CBM 64
GRAPHICS

EDITED BY NICK HAMPSHIRE

C.B.M.
64
GRAPHICS

NICK HAMPSHIRE

R.E. Stormer
Kleinejausrr.g
Huizer .

Duckworth

CONTENTS

Appendices

INTRODUCTION

This book is a compendium of graphics programs to assist the CBM 64 user to gain the greatest advantage from his computer. New users will find this book of great value once they understand Basic computing.

NICK HAMPSHIRE

4

COLOUR

COLOUR ON THE COMMODORE 64

INTRODUCTION

The Commodore 64 has the ability to display a wide range of colours, in any one of the three different colour modes. These modes range from the Standard 'normal' Display, as seen when the computer is on, through to what is known as the Multi-colour Mode. In Multicolour Mode, each character can be displayed with up to four colours. Finally, in the Extended Colour Mode, the user has the ability to control the background colour of any character, while using the foreground colour.

These three modes give the user a choice of different colours for visual displays. When not running a program, the user can – using the direct mode – access the first 8 colours by holding down the 'CONTROL' key and pressing any number from 1 through to 8. For example; by holding 'CONTROL' and depressing key '5' any characters thereafter typed will appear in purple on the screen.

The second 8 colours are accessed by holding down the 'Commodore Logo' key, and again, by depressing any key from 1 through to 8. To display a typed character on the screen, in one of the three shades of grey, hold down the 'Commodore Logo' key and key 5.

A full list of colours is as follows:

Pressing 'CONTROL' and:

1	2	3	4	5	6
Black	White	Red	Cyan	Purple	Green

7	8
Blue	Yellow

Pressing 'Commodore Logo' and:

1	2	3	4	5	6
Orange	Brown	Lt Red	Grey 1	Grey 2	Lt Green

7	8
Lt Blue	Grey 3

HOW COLOUR IS DISPLAYED

The Colour memory area is called from locations 55296 ($D800) to 56295 ($DBE7). Unlike most of the other memory areas in the Commodore 64, this one CANNOT be moved. These colour locations correspond to the usual screen memory locations of 1024 to 2023. To change the colour of a character stored in the top left hand corner of the screen, it is necessary to 'POKE 55296,X' — where X is a number from 0 to 15. As a further example, to change the colour of a character stored in the bottom right hand corner of the screen: 'POKE 56295,X'. As indicated, X must lie in the range of 0 to 15, and these numbers relate to the 16 colours mentioned earlier. Black is taken to be colour number 0, White colour number 1, and this continues up to the third shade of grey, which is colour number 15. In this way, to produce a PURPLE character in the top left hand corner of the screen, one has to 'POKE 55296,4'.

The above operation changes the foreground colour of the character. To change the background colour, memory locations 53280 and 53281 must be used, being the locations for the border colour and background colour 0. The background colours 1, 2 and 3 will be dealt with later.

'POKE-ing 53281' with any number between 0-15 mentioned earlier will give a border in the appropriate colour. For example, 'POKE 53280,4' will give a purple border. To change the background of the entire screen display, 'POKE 53281' with a number from 0 to 15. Using 'POKE 53280,4' and 'POKE 53281,4', both screen and border will appear in purple and the screen display will look bigger than it actually is.

In normal High Resolution Mode, control of colour on the screen is somewhat limited. Although it is possible to address each dot (pixel) in an 8 by 8 character space, if that dot is off, then the dot takes the existing background colour. If it is on, it takes the character colour assigned to that screen position. This may sound very good, but problems arise when two different coloured lines cross.

The Multi-colour Mode now becomes important. Each dot, in the 8 by 8 character space, can be any one of four colours: the screen colour, the background colour 0, the background colour 1 or 2 (memory locations 53282 and 53283), or the ordinary character colour. However, this gain is not achieved without the loss of some other function.

In the Multi-colour Mode, each dot is twice as wide as a dot in the high resolution mode.

Finally, the Extended Background Mode gives control over the background and foreground colour of each character. It is possible to have a red character with a white background on a black screen. This is achieved by control of the four memory locations from 53281 to 53284, each of which can hold any of the 16 colours. Control of the foreground colour is obtained by using the normal colour memory, which operates in exactly the same way as in normal Colour Mode.

As shown, the Extended Background may only be used when there is a loss of another function. Only the first 64 characters in the character ROM, or the first 64 in the programmable character set can now be used.

The Multi-colour and Extended Background Modes are referred to later in the text. Their precise working mechanism, and the method by which they can be accessed from Basic, will then be explained.

DESIGNING COLOUR DISPLAYS

Before producing any type of colour display, one needs to have an idea of what colours go well together, what restraints there are when using combinations of colours, how many colours per character space, and so on.

The following chart will give an idea of what colours are usable together on the same line of the TV, and what colours to avoid.

CHARACTER COLOUR

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0		*		*	*	+		*	*		*	*	*	*	*	*
1	*		*		*	*	*		+	*	+	*	*		*	*
2		*			+			*	*		*					+
3	*					+	*					+			+	
4	*		+													+
5	*	+		+								+		*		+
6	+	*		*										+	*	*
7	*		*				+		+	*	+	*	*			
8	+	*	*					*		*						+
9	*							*	*		*					*
10	+	+	*					+		*						+
11	*	*		+				*				*	*	+		*
12	*	*	+				+			+		*				*
13	*				*		+					*				
14	*	*		*			*						+			+
15	*	*	*		+	+	*			+	+	*	*		+	
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

↑

SCREEN COLOUR

Where '*' indicates 'Excellent'
Where '+' indicates 'Reasonable'
Where ' ' indicates 'Very Poor'

Before producing any colour displays, a glance at the above chart should give a fair idea of whether the display produced will look reasonable or not.

9

This, however, is using only one colour per character square What happens when using more?

MORE COLOURS PER CHARACTER SQUARE

Normally, one uses two colours per 8x8 character square: foreground and background. The foreground is set using the colour RAM in locations 55296 to 56295 to define the foreground colour of each particular square, and the screen background is set by 'POKE-ing' memory location 53281.

In Multicolour Mode, the screen's resolution is limited to 160 pixels by 200. In compensation for this, each dot in each 8x8 square can be any one of four colours: the screen colour, the colour in background registers 1 or 2, and the character colour.

To control the background colour for each character as well as the foreground colour, whilst leaving the overall screen background colour unaffected, enter what is known as Extended Colour Mode.

```
10 REM COLOUR DEMO
30 REM
40 VR=1024
50 KR=55296
60 PRINT"J"
70 POKE53281,1
80 FORI=VRTOVR+999
90 POKEI,160:NEXT
100 Z=KR:Q=KR+39
110 E=KR+960:R=KR+999
120 FORJ=0TO12
130 FORI=0TO19
140 X=INT(RND(1)*16)
150 POKEZ+J*41+I,X
160 POKEQ+J*39-I,X
165 C1=Z+J*41+I*40
166 IFC1>56295THEN175
170 POKEC1,X
175 C1=Q+J*39+I*40
176 IFC1>56295THEN190
180 POKEC1,X
190 POKEE-J*39+I,X
200 POKER-J*41-I,X
205 C1=E-J*39-I*40
206 IFC1<55296THEN215
210 POKEC1,X
215 C1=R-J*41-I*40
216 IFC1<55296THEN230
220 POKEC1,X
230 GETA$:IFA$<>""THEN270
240 NEXT
250 FORK=1TO200:NEXT
260 NEXT:GOTO120
270 PRINT"⊡⊡";:POKE53281,6
READY.
```

SETTING COLOUR FROM BASIC

Simple selection of colour in normal mode can be done via the keyboard, as described earlier, or from within a Basic program. The latter option will be considered here.

For the purpose of these examples, consider the screen RAM to be occupying memory locations 1024 to 2023. It can be moved about within RAM. Colour RAM, as seen, always occupies memory locations 55296 to 56295.

Each of these colour RAM locations is four bits wide, meaning that they can contain any integer from 0 to 15, as the Commodore 64 can use 16 possible colours.

The Memory location for altering the background colour of the screen is 53281, and by 'POKE-ing' this with any integer number from 0 to 15, this colour can be changed at will. The following short demonstration program runs through all the possible screen background colours:

```
10 PRINT "(CLR)" : REM CLEAR THE SCREEN
20 FOR I = 0 TO 15 : REM START OF LOOP
30 POKE 53281,I : REM CHANGE BACKGROUND COLOUR
40 FOR J = 1 to 1000 : NEXT J : REM DELAY
50 NEXT I : REM NEXT STEP IN LOOP
```

When run, this program will clear the screen, and then step through the 16 possible colours at the rate of about one per second. The delay loop in line 40 is simply to achieve this stepping process.

When finished, hit 'RUN-STOP/RESTORE' to get the display back to normal again.

To select the border colour, set memory location 53280 to any one of the 16 possible colours. One can alter the above program to set the screen background colour to white, and then step through all possible border colours.

```
10 POKE 53281,1 : REM SET SCREEN COLOUR TO WHITE
20 PRINT "(CLR)" : REM CLEAR THE SCREEN
30 FOR I = 0 TO 15 : REM START OF LOOP
40 POKE 53280,I : REM CHANGE BORDER COLOUR
50 FOR J = 1 TO 1000 : NEXT J : REM DELAY
60 NEXT I : REM NEXT STEP IN LOOP
```

When run, this program will set the screen colour to white, clear the screen, and then step through each border colour at the rate of about one per second.

Both these changes can combine to produce a bewildering variety of screen/border colour combinations. In the following program, no delay loop has been inserted (although a delay loop can be inserted by perhaps copying line 50 from the last program), and instead, the program just flashes through all the changes.

```
10 PRINT "(CLR)" : REM CLEAR THE SCREEN
20 FOR I = 0 TO 15 : REM SET UP SCREEN CHANGE LOOP
30 FOR J = 0 TO 15 : REM SET UP BORDER CHANGE LOOP
40 POKE 53280,J : REM CHANGE BORDER COLOUR
50 NEXT J : REM NEXT STEP IN BORDER LOOP
60 POKE 53281,I : REM CHANGE SCREEN COLOUR
70 NEXT I : REM ONCE BORDER FINISHED, CHANGE SCREEN
AGAIN
```

When run, the program clears the screen, steps through all sixteen border colours, then changes the screen colour, steps through the border colours again, and so on.

As usual, hitting 'RUN-STOP/RESTORE' will get the screen display back to normal.

The colour RAM is closely linked to the screen RAM. By changing any byte of the colour RAM, typing in an integer from 0 to 15 will change the colour of the character at that particular location on the screen. For example, if you have a white screen and a blue letter A at screen location 1024, i.e. at the top left hand corner of the screen, to change that to a purple A, type POKE 55296, 4.

The following program fills every character space on the screen with a random letter, and then changes that letter to a random colour. Using a black border and a white screen, the resulting display may not please, but at least gives some idea of how the colour RAM works in conjunction with screen RAM.

```
10 POKE 53280,0 : REM SET BLACK BORDER
20 POKE 53281,1 : REM SET WHITE SCREEN
30 FOR I = 1024 TO 2023 : REM SET UP SCREENFILL LOOP
40 POKE I, INT(RND(0.5)*25+1): REM POKE RANDOM LETTER
ONTO SCREEN
50 NEXT I : REM NEXT STEP IN SCREENFILL LOOP
```

```
60 FOR J = 55296 TO 56295 : REM SET UP COLOUR CHANGE
LOOP
70 POKE J, INT(RND(1)*16) : REM POKE RANDOM COLOUR
80 NEXT J : REM NEXT STEP IN COLOUR CHANGE LOOP
```

When run, hit RUN-STOP/RESTORE to get the screen display back to normal again.

One final point. Colour codes, or border changes, or whatever, can of course be input into a program and acted upon. The following short program asks you to type in a colour for the border display, and changes the border accordingly. Typing 16 exits the program.

```
10 PRINT "(CLR)" : REM CLEAR THE SCREEN
20 INPUT A : REM INPUT A NUMBER
30 IF A<0 OR A>16 THEN 20 : REM DON'T ACCEPT INVALID
INPUT
35 IF A=16 THEN END
40 POKE 53280,A : REM CHANGE BORDER COLOUR AS
REQUESTED
50 GO TO 20 : REM BACK AND DO IT AGAIN
```

14

SETTING REGISTERS FOR MULTI-COLOUR MODE

We have discussed the advantages and disadvantages of programming in one or other of the various colour modes available.

Let us now consider the Multi-colour Mode, the methods of displaying characters in that mode, the sort of displays that can be achieved, and how to interleave this with normal high resolution displays.

First, what, precisely, is the Multi-colour Mode?

In normal high resolution mode, you can control a screen definition area of 320 x 200 pixels, and each of these pixels can have two possible colours. Either the upper 4 bits of the video RAM for the points, or the lower 4 bits for the background (see section on Hi-Resolution Graphics).

Although this does give a fairly fine control over video output, it is not without its limitations. As seen before, certain character colours do not go well together, and this can lead to problems in high resolution colour displays. Having only two colours per character space, one of those must be the overall screen background colour.

To overcome this, enable Multi-colour Mode. This gives each dot within a character square the ability to be displayed in one of any four colours. The screen colour, or background colour zero (memory location 53281), background colours one or two (locations 53292 and 53283), or the character colour accessed from the colour.

It is necessary to make one sacrifice, in terms of horizontal resolution, as each dot is now double its original width, or, to put it another way, the user could have control over a resolution area of 160 x 200 dots.

To enable Multi-colour Mode, set bit 4 of the Vic II control register, which sits at memory location 53270 ($D016) to contain a 1, which is done with the following POKE command:

POKE 53270, PEEK(53270) OR 16

To turn it off, the same bit must be set to 0 (zero) which is done

by:

POKE 53270, PEEK (53270) AND 239

Once you have decided to have a multi-colour display, it is the bit pattern that goes to make up the character there, determining what colour(s) that character will be displayed in. Sticking to letter 'A', which in normal mode is made up as follows:

LETTER	BIT DISPLAY
**	00011000
****	00111100
** **	01100110
******	01111110
** **	01100110
** **	01100110
** **	01100110
	00000000

In normal character mode, or standard high resolution mode, the screen background colour shows where there is a 'O' and the character colour selected for that square where there is a '1'. In multi-colour mode, in which individual dots cannot be addressed and, instead, pairs of dots are addressed, the resultant display becomes:

LETTER	BIT DISPLAY
XXYY	00 01 10 00
ZZZZ	00 11 11 00
XXYYXXYY	01 10 01 1U
XXZZZZYY	01 11 11 10
XXYYXXYY	01 10 01 10
XXYYXXYY	01 10 01 10
XXYYXXYY	01 10 01 10
	00 00 00 00

Wherever there is an XX, background colour 'one' will be displayed. YY denotes background colour 'two' and ZZ denotes the character colour. Spaces will be displayed as the screen background colour : four colours per character square.

In other words, bit pair 00 represents the screen colour, 01 background colour one, 10 background colour two, and 11 the character colour. To recap, these are memory locations 53281, 53282, 53283 and the colour RAM respectively.

16

Thus, to create a multi-colour display, first of all turn that mode on, then simply decide what colour everything is to be. The following program should serve as a demonstration of this:

```
10 PRINT "(CLR)XXXXXXXXXX" : REM CLEAR THE SCREEN,
THEN PRINT 10 X'S
20 POKE 53281,7 : REM SET BACKGROUND COLOUR ZERO TO
YELLOW
30 POKE 53282,2 : REM SET BACKGROUND COLOUR ONE TO
RED
40 POKE 53283,14: REM SET BACKGROUND COLOUR TWO TO
LIGHT BLUE
50 POKE 53270, PEEK (53270) OR 16 : REM TURN MULTI-COLOUR
MODE ON
60 A=55296 : REM START OF COLOUR RAM
70 FOR I = 0 TO 9 : REM START OF LOOP
80 POKE A+I,14 : REM BLUE 'CLAUSE IN MULTI-COLOUR MODE
90 NEXT I : REM NEXT STEP IN LOOP
```

This program clears the screen and prints up 10 successive X's, sets the various background colours, and then changes the colour of the X's at the top of the screen.

Of course, when in Multi-colour Mode, one is no longer putting integer numbers in memory locations in colour RAM, but actually referencing the bit registers in those locations. Changing one register can instantly change every dot drawn in a particular colour. Good for fast action arcade graphics!!

The following program will show how this works: changing background colour register number 'one' changes everything from (in this case) red to various random colours.

```
10 PRINT "(CLR)" : REM CLEAR THE SCREEN
20 POKE 53270, PEEK (53270) OR 16 : REM TURN ON
MULTI-COLOUR MODE
30 POKE 53282,10 : REM BACKGROUND COLOUR ONE SET TO
RED
40 FOR I = 1 TO 999 : REM START OF LOOP
50 PRINT 'X'; : REM PRINT 'X' CHARACTER ON SCREEN
60 NEXT I : REM 999 TIMES I.E. FILL UP THE SCREEN
70 A = INT(RND(0.5)*16) : REM PRODUCE A NUMBER FROM 0 TO
15
80 POKE 53282,A : REM CHANGE BACKGROUND COLOUR 1
```

```
RANDOMLY
90 FOR J = 1 TO 1000 : NEXT : REM WAIT FOR ABOUT 1 SECOND
100 GOTO 70 : REM CHANGE COLOUR AGAIN
```

This program clears the screen, prints 999 X's with red backgrounds, then changes it all to a random background colour. After a delay of approximately one second, the display changes again and carries on doing so until the 'RUN-STOP' key is pressed. Pressing 'RUN-STOP/RESTORE' should set everything back to normal again.

Knowing now how to create multi-colour displays, and how, by selective use of colour RAM, to have a mixture of multi-colour and normal high resolution characters on display leaves the third colour option, namely Extended Colour Mode.

USING EXTENDED COLOUR MODE

This is the third colour mode of display, one which cannot be used with Multi-colour Mode graphics as it affects the whole screen. Thus it must be used with care, and for special circumstances only.

It gives the ability to control not only the background colour of the screen, but also the background (and foreground) colour of each character displayed on the screen. For example, a red character with a yellow background displayed on a blue screen.

Familiar registers are used to store information about the chosen colours, namely background registers 0, 1, 2 and 3 (memory locations 53281 to 53284 respectively), with each register capable of being set to any one of the usual 16 colours. Not going into multi-colour mode here, the colours all have their true displays, i.e. 11 is the first shade of grey rather than cyan.

To select the foreground colour of a character, use the usual colour RAM to switch from one colour to another.

There is one limitation in using extended background colour mode (to give it its full title) and this is that only the first 64 characters stored in ROM, or the first 64 characters in the programmable character set, can be accessed since two of the bits of the character code are used to select the background colour.

To give an example, 'POKE 1024,24' and one would expect a normal X to appear on the screen. 'POKEing' 1024 with 88 would usually bring up a reverse field X, but with Extended Colour Mode on, this is not the case, giving instead the same X as before, the most awkward to implement. Still, a certain amount of experimentation should soon see some very colourful displays.

To affect the colour of a character 'POKEd' onto the screen with a character code range of 0 to 63, alter background colour register 0, the range of 64 to 127 for register 1, 128 to 191 for register 2, and finally 192 to 255 is changed by background colour register 3.

Before doing this, one must turn the Extended Background Colour Mode 'on'. This is achieved with the following command:

POKE 53265, PEEK (53265) OR 64

and turned off with:

POKE 53265, PEEK (53265) AND 191

This sets bit 6 of the Vic II register at 53265 on and off (or to 1 and 0) respectively.

The following program should serve to show how all of this might work in practice.

```
10 PRINT "(CLR)" : REM CLEAR THE SCREEN
20 POKE 53265, PEEK (53265) OR 64 : REM ENABLE EXTENDED
MODE
30 POKE 53280,1 : REM WHITE BORDER
40 POKE 53281,0 : REM BLACK SCREEN
50 POKE 1024,44 : REM PUT AN X IN TOP LEFT HAND CORNER OF
SCREEN
60 POKE 55296,1 : REM CHANGE IT TO WHITE
70 POKE 53282,7 : REM SET BACKGROUND COLOUR 1 TO
YELLOW
80 POKE 1024,88 : REM WHITE X WITH YELLOW BACKGROUND
ON BLACK SCREEN
90 etc. . . .
```

So, simply by changing the appropriate background and character colours the user may have a bewildering variety of colours on display at the same time.

```
10 REM YOU CAN REMOVE ALL REMS
20 REM * SET TO LOWER CASE AND SET VARIABLES
30 POKE53272,23:ZZ=53281:X=54272
40 PRINT"◻       ◪********   DEMO   *******"
50 REM * SWITCH TO EXTENDED COLOUR MODE
60 POKE53265,PEEK(53265)OR64
70 FORD=1TO500:NEXT:REM * DELAY LOOP
80 REM A2$="SHIFT SPACES'FOUR'SHIFT SPACES"
90 REM A4$="SHIFT SPACES'CHANGING COLOURS'SHIFT SPACES"
100 A1$="COMMODORE COLOUR POWER ON YOUR 64"
110 A2$="          ⌐ ∠              "
120 A3$="◪          BACKGROUND                ■"
130 A4$="◪      ─ ◆∕⌐∖∕⌐ ─⊓⌐∠◆        ■"
140 REM * SET BACKGROUND COLOUR REGISTERS
150 POKEZZ,9
160 POKEZZ+1,4
170 POKEZZ+2,7
180 POKEZZ+3,0
190 FORD=1TO1000:NEXT:REM * DELAY LOOP
200 REM * LOOP TO SET BACK AND FORE COLOURS
210 FORI=0TO100
220 REM * SET CURRENT CHARACTER COLOUR
230 POKE646,(I+4)AND15
240 IF(IAND3)=0THENPRINTA1$
250 IF(IAND3)=1THENPRINTA2$
260 IF(IAND3)=2THENPRINTA3$
270 IF(IAND3)=3THENPRINTA4$
280 POKEZZ,I
290 POKEZZ+1,I+2
300 POKEZZ+2,I+3
310 POKEZZ+3,I+1
320 FORD=1TO100:NEXT
330 NEXT
340 FORG=0TO3:POKEZZ+G,15:NEXT
350 FORD=0TO3000:NEXT
360 FORI=0TO70
370 Z=ZZ+(IAND3):REM * DRAWS MOVING BARS
380 POKEZ,12:REM * DRAW BAR
390 FORD=1TO200:NEXT
400 POKEZ,15:REM * UNDRAW BARS
410 NEXT
420 REM * RESET COLOUR AND EXIT
430 POKE53272,21:POKE53265,PEEK(53265)AND191
440 END
READY.
```

```
1000 REM *************************************
1010 REM * LOADER FOR A MULTI-COLOUR HI-RES   *
1020 REM * MACHINE CODE ROUTINE TO SET UP THE *
1030 REM * HI-RES SCREEN AND PLOT A POINT USING *
1040 REM * THE COLOURS AVAILABLE.             *
1050 REM *                                    *
1060 REM * THE ROUTINES ARE CALLED THUS:      *
1070 REM *    SC=SCREEN COLOUR                *
1080 REM *    BC=BORDER COLOUR                *
1090 REM *    SYS(28672),SC,BC                *
1100 REM * THIS IS TO SET UP THE HI-RES SCREEN *
1110 REM *************************************
1120 REM *    PC=POINT COLOUR :FROM 0-15      *
1130 REM *    BR=BRUSH NUMBER :FROM 0-3       *
1140 REM *                    :0=BACKGROUND COL. *
1150 REM *    POKE 828,BR                     *
1160 REM *    POKE 829,PC                     *
1170 REM *    SYS(28928),X,Y                  *
1180 REM * THIS PLOTS A POINT ON THE SCREEN AT *
1190 REM * COORDINATES X,Y WHERE 0,0 IS TOP LEFT*
1200 REM *************************************
2000 I=28672:T=0:ER=0
2010 READ A
2020 IF A>255 THEN 2500
2030 IF A=-1 THEN 3000
2040 POKE I,A:T=T+A
2050 I=I+1
2060 GOTO 2010
2500 READ A$
2510 IF T=A THEN PRINT"LINES "A$" O.K.":GOTO 2530
2520 PRINT"LINES "A$" ENTERED INCORRECTLY   "T,A:ER=1
2530 T=0:GOTO 2010
3000 I=28928:T=0
3010 READ A
3020 IF A>255 THEN 3500
3030 IF A=-1 THEN 4000
3040 POKEI,A:T=T+A
3050 I=I+1
3060 GOTO 3010
3500 READ A$
3510 IF A=T THEN PRINT"LINES "A$" O.K.":GOTO 3530
3520 PRINT"LINES "A$" ENTERED INCORRECTLY   "T,A:ER=1
3530 T=0:GOTO 3010
4000 IF ER=1 THEN END
4010 POKE51,0:POKE52,112
4020 POKE55,0:POKE56,112
4030 CLR:NEW
20000 DATA32,253,174,32,235,183,138
20010 DATA141,32,208,165,20,141,33
20020 DATA208,32,49,112,32,88,112
```

```
20030 DATA32,127,112,173,17,208,9
20040 DATA32,141,17,208,173,24,208
20050 DATA9,8,141,24,208,173,22
20060 DATA208,9,16,141,22,208,96
20070 DATA160,0,169,64,133,87,169
20080 DATA63,133,88,169,0,145,87
20090 DATA165,87,240,5,198,87,76
20095 DATA 7511,"20000-20090"
20100 DATA59,112,198,88,165,88,201
20110 DATA31,240,7,169,255,133,87
20120 DATA76,59,112,96,160,0,169
20130 DATA231,133,87,169,7,133,88
20140 DATA169,0,145,87,165,87,240
20150 DATA5,198,87,76,98,112,198
20160 DATA88,165,88,201,3,240,7
20170 DATA169,255,133,87,76,98,112
20180 DATA96,160,0,169,231,133,87
20190 DATA169,219,133,88,169,0,145
20195 DATA 8541,"20100-20190"
20200 DATA87,165,87,240,5,198,87
20210 DATA76,137,112,198,88,165,88
20220 DATA201,215,240,7,169,255,133
20230 DATA87,76,137,112,96
20240 DATA 3461,"20200-20230",-1
30000 DATA32,253,174,32,235,193,165
30010 DATA20,133,89,165,21,133,90
30020 DATA138,201,200,144,2,169,199
30030 DATA133,91,165,90,201,1,144
30040 DATA16,208,6,165,89,201,64
30050 DATA144,8,169,1,133,89,169
30060 DATA63,133,90,165,89,41,7
30070 DATA133,94,169,7,56,229,94
30080 DATA133,94,70,94,6,94,165
30090 DATA94,24,105,1,133,95,165
30095 DATA 7708,"30000-30090"
30100 DATA94,240,9,168,169,1,10
30110 DATA136,208,252,240,2,169,1
30120 DATA133,94,165,95,240,9,168
30130 DATA169,1,10,136,208,252,240
30140 DATA2,169,1,133,95,169,0
30150 DATA133,92,133,88,133,87,165
30160 DATA91,41,7,133,93,165,91
30170 DATA74,74,74,133,91,160,5
30180 DATA24,10,38,88,136,208,249
30190 DATA133,87,165,91,160,3,24
30195 DATA 7667,"30100-30190"
30200 DATA10,136,208,251,133,91,24
30210 DATA101,87,133,91,165,88,105
30220 DATA0,133,92,160,3,24,70
30230 DATA90,102,89,136,208,248,165
30240 DATA89,133,87,165,90,133,88
```

```
30250 DATA160.3.24.6.87.38.88
30260 DATA136.208.248.160.8.24.165
30270 DATA91.101.87.133.87.165.92
30280 DATA101.88.133.88.136.208.240
30290 DATA24.165.93.101.87.133.87
30295 DATA 7723."30200-30290"
30300 DATA169.0.101.88.24.169.32
30310 DATA101.88.133.88.24.165.89
30320 DATA101.91.133.91.169.0.101
30330 DATA92.133.92.173.60.3.201
30340 DATA0.240.13.201.1.240.32
30350 DATA201.2.240.74.201.3.240
30360 DATA104.96.160.0.165.94.73
30370 DATA255.133.94.165.95.73.255
30380 DATA133.95.177.87.37.94.37
30390 DATA95.145.87.96.160.0.165
30395 DATA 7569."30300-30390"
30400 DATA95.73.255.133.95.177.87
30410 DATA5.94.37.95.145.87.169
30420 DATA4.24.101.92.133.92.14
30430 DATA61.3.14.61.3.14.61
30440 DATA3.14.61.3.177.91.41
30450 DATA15.24.109.61.3.145.91
30460 DATA96.160.0.165.94.73.255
30470 DATA133.94.177.87.37.94.5
30480 DATA95.145.87.24.169.4.101
30490 DATA92.133.92.177.91.41.240
30495 DATA 6023."30400-30490"
30500 DATA24.109.61.3.145.91.96
30510 DATA160.0.177.87.5.94.5
30520 DATA95.145.87.169.216.24.101
30530 DATA92.133.92.173.61.3.145
30540 DATA91.96
30550 DATA 2780."30500-30540".-1
READY.
```

```
1000 REM *********************************
1010 REM * THIS PROGRAM IS A SKETCHING MULTI- *
1020 REM * COLOUR PALLET. IT ALLOWS YOU TO PLOT *
1030 REM * POINTS ON TO THE SCREEN USING THE *
1040 REM * CURSOR KEYS IN MULTI COLOUR MODE. *
1050 REM * THERE ARE TWO SECTIONS OF THE PROGRAM*
1060 REM * THEY ARE AS FOLLOWS: *
1070 REM *    COMMAND SECTION: IN THIS SECTION, *
1080 REM * THE USER CAN DEFINE WHICH COLOUR TO *
1090 REM * WHICH 'PAINT BRUSH'. EXIT PROGRAM. OR*
1100 REM * CHOOSE TO PLOT POINTS. *
1110 REM *    DEFINE COLOURS IS SELECTED BY *
1120 REM * PRESSING 'F1'. MOVE THE CURSOR OVER *
1130 REM * THE PALLET USING THE KEYS *
1140 REM * CURSOR RIGHT TO MOVE IT TO THE RIGHT *
1150 REM * CURSOR DOWN  TO MOVE IT TO THE LEFT *
1160 REM * WHEN THE CURSOR IS IN POSITION. *
1170 REM * PRESSING 'F3' SELECTS THAT COLOUR. *
1180 REM * THEN ONE OF KEYS 1.2.3 ARE PRESSED TO*
1190 REM * SET THAT COLOUR TO THAT BRUSH. *
1200 REM *    THE SECOND SECTION IS THE PLOT *
1210 REM *    TO PLOT POINTS, PRESS THE KEY 'P' *
1220 REM * POINTS SECTION. *
1230 REM * A BRUSH MAY BE SELECTED AT ANY TIME *
1240 REM * BY PRESSING THE CPORRESPONDING KEY. *
1250 REM * THE CURSOR KEYS MOVE THE HIRES CURSOR*
1260 REM * TO EXIT PLOT MODE. PRESS THE RETURN *
1270 REM * KEY. *
1280 REM *    TO EXIT THE PROGRAM COMPLTELY. *
1290 REM * PRESS RETURN WHEN IN COMMAND MODE *
1300 REM * *
1310 REM * THIS PROGRAM USES THE MULTI-COLOUR *
1320 REM * HI-RES MACHINE CODE ROUTINES. *
1330 REM *********************************
1340 B=8192
1350 INPUT":SCREEN COLOUR";N
1360 SYS28672,N,N
1370 DEFFNP(ZZ)=B+INT(X/8)*8+INT(Y/8)*320+(YAND7)
1380 DEFFNK(ZZ)=55296+INT(X/8)+INT(Y/8)*40
1390 GOSUB 1970:GOSUB 1870
1400 GETA$:IFA$=""THEN1400
1410 IF A$="■" THEN 1490
1420 IF A$=CHR$(13) THEN 1950
1430 IF A$="0" THEN BR=0
1440 IF A$="1" THEN BR=1
1450 IF A$="2" THEN BR=2
1460 IF A$="3" THEN BR=3
1470 IF A$="P" THEN 1670
1480 GOTO 1400
1490 I=0
1500 Y=192:X=I*16+40
```

25

```
1510 POKE FNK(0),N
1520 FOR J=1 TO 50
1530 GET A$:IF A$<>""THEN 1550
1540 NEXT J
1550 POKE FNK(0),I
1560 IF A$="N"THEN I=I-1
1570 IF A$="M"THEN I=I+1
1580 IF A$="▆"THEN CL=I:POKE FNK(0),I:GOTO 1620
1590 IF I<0 THEN I=0
1600 IF I>15 THEN I=15
1610 GOTO 1500
1620 GET A$:IF A$="" THEN 1620
1630 IF A$="1" THEN COL(1)=CL
1640 IF A$="2" THEN COL(2)=CL
1650 IF A$="3" THEN COL(3)=CL
1660 GOSUB 1870:GOTO 1400
1670 X=XX:Y=YY
1680 POKE828,0:SYS28928,X,Y
1690 GET A$
1700 POKE828,BR:POKE829,COL(BR):SYS28928,X,Y
1710 IF A$="" THEN 1680
1720 IF A$="0" THEN BR=0:GOTO1680
1730 IF A$="1" THEN BR=1:GOTO1680
1740 IF A$="2" THEN BR=2:GOTO1680
1750 IF A$="3" THEN BR=3:GOTO1680
1760 IF A$="M" THEN Y=Y+1
1770 IF Y>183 THEN Y=183
1780 IF A$="⊐" THEN Y=Y-1
1790 IF Y<0 THEN Y=0
1800 IF A$="M" THEN X=X+2
1810 IF X>319 THEN X=319
1820 IF A$="M" THEN X=X-2
1830 IF X<0 THEN X=0
1840 IF A$=CHR$(13) THEN XX=X:YY=Y:GOTO 1400
1850 POKE828,BR:POKE829,COL(BR):SYS28928,X,Y
1860 GOTO 1680
1870 FOR I=1 TO 3
1880 Y=184:X=16*I+49
1890 POKEFNK(0),COL(I)
1900 FOR J=0 TO 7
1910 POKEFNP(0),255
1920 Y=Y+1:NEXT J
1930 NEXT I
1940 RETURN
1950 POKE53265,27:POKE53272,21
1960 POKE53270,200:POKE53281,6:PRINT"⊐":END
1970 FORI=0TO15
1980 X=16*I+40
1990 Y=192:POKEFNK(0),I
2000 FORJ=0 TO 7
2010 POKEFNP(0),255
```

```
2020 Y=Y+1
2030 NEXT J
2040 NEXT I
2050 RETURN
READY.
```

This program will sometimes appear to be faulty. It is in fact working, but the pixel colour needs to be changed so that it is visible (i.e. press "1").

GRAPHICS

HIGH RESOLUTION GRAPHICS

Given the amount of memory 'on board' the 64, it is a pity that there are no High-Resolution Graphics commands directly available to the user. In this section is a basic loader for a standard Hi-Resolution graphics package. The package consists of four routines in machine code which, when called with the appropriate parameters 'POKE'd' into the 64 and the correct 'SYS' call, will:

(1) Set up the hi-res screen,

(2) plot a single point,

(3) plot a line between two points,

(4) and plot characters on the screen at any co-ordinate.

PRINCIPLES OF HI-RES GRAPHICS

The rest of the section takes a look at how the hi-res routines are done. This will be split up into the four routines as mentioned above.

(1) Set up the hi-res screen:

This is the easiest of the routines, it can be divided into three parts:

(a) The first part sets the video memory to the desired colour combination. The hi-res colours are taken from the character displayed on the video screen, the screen colour is taken from the lower four bits, and the point colour from the upper four bits of the character stored in the RAM. Thus, if character 1('A') is used then the displays at that character position would be a white background with black points. To determine the character number from the required colours, if we let SC=the screen colour, and PC=the point colour, then the character number to be 'POKE'd' would be SC+16*PC.

(b) The second part clears the hi-res area of memory. This is an 8K block where points will be plotted. Using back 0, this memory block is from 8192 ($2000) to 16191 ($3F3F). To clear this a loop for 8000 will be needed and zero will have to be 'POKE'd' into each location in the hi-res area.

(c) The final part sets up the registers to intialise the Hi-Res mode.

There are two 'POKE's' required to do this, these are: POKE 53265,PEEK(53265)OR32 which puts the 64 into bit mapped mode and POKE 53272,PEEK(53272)OR8 which places the start of bit map memory at 8192.

BASIC VERSION OF SET UP ROUTINE

```
10 SC=12:PC=6
20 FOR I=1024 TO 2023
30 POKE I,SC+PC*16
40 NEXT
50 FOR I=8192 TO 16191
60 POKE I,0
70 NEXT I
80 POKE 53265,PEEK(53265)OR32
90 POKE 53272,PEEK(53272)OR8
READY.
```

In Multi-Colour Mode, there is a choice of four colours for each character. The video memory contains information on two of the colours, the colour nybble memory indicates the third colour and the screen colour register (53281) is the screen colour. An extra 'POKE' is required to select Multi-colour Mode, this is: POKE 53270,PEEK(53270)OR16.

(2) Plot a point at the desired X,Y co-ordinate.

This routine involves calculating the location in the bit mapped screen memory where the point is to be plotted, the bit required and then 'POKEing' to that byte to turn on the calculated bit. Although the screen and colour combinations have been entered in the set up routine, other colours may be required for plotting, rather than the point colour previously set up. This is done by calculating the byte on the video memory where the point occurs and changing the lower four bits to the required colour vehicle. If the lower four bits are changed, the character position containing the point will have a different background colour. A last note on this routine is that the convention of 0,0 being TOP left corner instead of bottom left corner has been used for ease of calculation.

All calculations are shown in the following basic listing. Note that this program will not work alone, it has been written for ease of understanding of the techniques.

```
1000 DEF FNP(ZZ)=8192+INT(X/8)*8+INT(Y/8)*320+(YAND7)
1010 DEF FNC(ZZ)=1024+INT(X/8)+INT(Y/8)*40
1020 X=<X COORD>:Y=<Y COORD>
1030 AD=<ADD OR DELETE POINT>
1040 PC=<POINT COLOUR>
1050 IF X<0 OR X>319 THEN 1160
1060 IF Y<0 OR Y>199 THEN 1160
1070 IF AD=1 THEN 1110
1080 BI=7-(XAND7)
1090 POKE FNP(0),PEEK(FNP(0))OR2↑BI
1100 GOTO 1130
1110 BI=7-(XAND7)
1120 POKE FNP(0),PEEK(FNP(0))AND255-(2↑BI)
1130 SC=PEEK(FNC(0))AND15
1140 COL=SC+PC*16
1150 POKE FNC(0),COL
1160 RETURN
READY.
```

In Multi-Colour Mode, each point takes up two bits of the bit map like this:

BITS   COLOUR FROM

00   Screen colour register (53281) (no point),
01   Upper 4 bits of video memory,
10   Lower 4 bits of video memory,
11   Colour nybble memory.

This means that the byte calculation stays the same but the bit calculation is a little different. Instead of the bit calculation being 7-(XAND7), the easiest way is to calculate the lower of the two bits and then plot each point. The calculation is BI=INT(7-(XAND7)/2)*2 and the two bits to be plotted are BI and BI+1. Because of the double byte plotting, the x resolution has been halved from 320 pixels to 160 but the y resolution stays at 200 pixels.

There are two more routines, they plot a line and plot a character. Both routines use the point plot routine to display on the screen but each has useful calculations that must be mentioned.

(3) Plot a line between two sets of co-ordinates.

```
2000 X1=<X START OF LINE>:Y1=<Y START OF LINE>
2010 X2=<X END OF LINE>:Y2=<Y END OF LINE>
2020 AD=<ADD OR DELETE LINE>:FC=<COLOUR OF LINE>
2030 XD=X2-X1
2040 YD=Y2-Y1
2050 A0=1:A1=1
2060 IF YD<0 THEN A0=-1
2070 IF XD<0 THEN A1=-1
2080 XE=ABS(XD):YE=ABS(YD):D1=XE-YE
2090 IF D1>=0 THEN 2130
2100 S0=-1:S1=0:LG=YE:SH=XE
2110 IF YD>0 THEN S0=1
2120 GOTO 2150
2130 S0=0:S1=-1:LG=XE:SH=YE
2140 IF XD>=0 THEN S1=1
2150 TT=LG:TS=SH:UD=LG-SH:CT=SH-LG/2
2160 D=0
2170 X=X1:Y=Y1
2180 GOSUB 1050
2190 IF CT>0 THEN 2220
2200 CT=CT+TS:X=X+S1:Y=Y+S0
2210 GOTO 2230
2220 CT=CT-UD:X=X+A1:Y=Y+A0
2230 TT=TT-1
2240 IF TT<=0 THEN 2260
2250 GOTO 2180
2260 RETURN
READY.
```

The line plot routine is no different whether in Standard Hi-Res Mode or in Multi-Colour Mode, but the character plot routine must be enlarged on the x axis by changing line 330 from X1=X1+1 to X1=X1+2.

(4) Plot a character at x,y screen co-ordinates with the top left corner of the character at these co-ordinates. This routine converts the ASCII character values into the equivalent 'POKE' value, reads the character from the character ROM, and plots each point of the character onto the screen. It must be noted, however, that none of the Commodore graphics characters or the lower case characters are available. Other than that, any letter, number, or other standard character is available.

```
4000 DEF FNP(ZZ)=8192+INT(X/8)*8+INT(Y/8)*320+(YAND7)
4010 A$="<STRING>":X=<X COORD>:Y=<Y COORD>
4020 XX=<X STEP>:YY=<Y STEP>
4030 FOR I=1 TO LEN(A$)
4040 B$="MID$(A$,I,1):B=ASC(B$)
4050 Y1=Y
4060 IF B<32 THEN NEXT I:GOTO 390
4070 IF B>31 AND B<64 THEN A=B
4080 IF B<63 AND B>96 THEN A=B-64
4090 IF B>95 THEN NEXT I:GOTO 390
4100 A=A*8:GEN=A+53248
4110 FOR J=0 TO 7
4120 X1=X
4130 POKE 56334,PEEK(56334)AND254
4140 POKE 1,PEEK(1)AND251
4150 Z=PEEK(GEN(J))
4160 POKE 1,PEEK(1)OR4
4170 POKE 56334,PEEK(56334)OR1
4180 PN%=128
4190 IF (PN%ANDZ)=0 THEN 4210
4200 POKE FNP(0),PEEK(FNP(0))OR2↑(7-(X1AND7))
4210 PN%=INT(PN%/2)
4220 IF PN%<1 THEN 4250
4230 X1=X1+1
4240 GOTO 4190
4250 Y1=Y1+1
4260 NEXT J
4270 X=X+XX:Y=Y+YY
4280 NEXT I
4290 RETURN
READY.
```

```
10 REM ***************************************************
20 REM * THIS IS THE COMPLETE HI-RES PACKAGE.     *
30 REM * IT COMPRISES OF ROUTINES TO              *
40 REM * 1) SET UP GRAPHICS MODE.                 *
50 REM * 2) PLOT A POINT AT THE DESIRED X.Y       *
60 REM *     COORDINATE ON THE SCREEN WHERE       *
70 REM *     TOP LEFT COORDINATES ARE 0.0.        *
80 REM * 3) DRAW A LINE BETWEEN TWO POINTS ON     *
90 REM *     SCREEN.                              *
100 REM* 4) PLOT UP CHARACTERS ON THE SCREEN      *
110 REM*    AT THE DESIRED X.Y COORDINATES OF     *
120 REM*    THE TOP LEFT POINT OF THE CHARACTER.  *
130 REM***************************************************
140 REM* 1) THIS ROUTINE IS SPLIT UP INTO THREE   *
150 REM*    PARTS. THE FIRST CLEARS THE 8K BIT    *
160 REM*    MAP MEMORY. THE SECOND SETS THE       *
170 REM*    COLOUR COMBINATIONS FOR SCREEN AND    *
180 REM*    POINT COLOUR. AND THE THIRD SETS THE  *
190 REM*    REGISTERS FOR BIT MAP MODE.           *
200 REM*    THE ROUTINE IS CALLED BY              *
210 REM*     POKE 89.SC+16*PC (SC=SCREEN COLOUR)  *
220 REM*                     (PC=POINT COLOUR)    *
230 REM*     SYS(28672).                          *
240 REM*    THE FIRST TWO PARTS MAY BE CALLED ON  *
250 REM*    THEIR OWN WITH                        *
260 REM*    A) SYS(28704)                         *
270 REM*    B) THE ABOVE POKE FOLLOWED BY         *
280 REM*       SYS(28743)                         *
290 REM***************************************************
300 REM* 2) THIS ROUTINE PLOTS A POINT ON THE     *
310 REM*    SCREEN WITH THE DESIRED COLOUR        *
320 REM*    IT IS CALLED BY                       *
330 REM*     POKE 89.X-INT(X/256)*256             *
340 REM*     POKE 90.INT(X/256)                   *
350 REM*     POKE 91.Y                            *
360 REM*     POKE 784.AD    (AD =1 FOR PLOT. )    *
370 REM*                    (   =2 FOR DELETE)    *
380 REM*     POKE 879.PC                          *
390 REM*     SYS(28782)                           *
400 REM***************************************************
410 REM* 3) THIS ROUTINE DRAWS A HI-RESOLUTION    *
420 REM*    LINE ON THE SCREEN BETWEEN TWO SETS   *
430 REM*    OF COORDINATES IN THE DESIRED COLOUR  *
440 REM*    IT IS CALLED BY                       *
450 REM*     POKE 828.X1-INT(X1/256)*256          *
460 REM*     POKE 829.INT(X1/256)                 *
470 REM*     POKE 830.Y1:POKE 831.0               *
480 REM*     POKE 832.X2-INT(X2/256)*256          *
490 REM*     POKE 833.INT(X2/256)                 *
500 REM*     POKE 834.Y2:POKE 835.0               *
510 REM*     POKE 786.AD                          *
```

```
520 REM*      POKE 879.PC                           *
530 REM*      SYS(29158)                            *
540 REM******************************************
550 REM* 4) ROUTINE TO PLOT A CHARACTER ON THE  *
560 REM*    SCREEN WITH X.Y THE TOP LEFT CORNER *
570 REM*    OF THE CHARACTER.                   *
580 REM*    IT IS CALLED BY                     *
590 REM*    FOR I=1 TO LEN(B$)                  *
600 REM*    A$=MID$(B$.I.1)                     *
610 REM*    POKE 866.X-INT(X/256)*256           *
620 REM*    POKE 867.INT(X/256)                 *
630 REM*    POKE 868.Y:POKE 869.0               *
640 REM*    POKE 870.ASC(A$)                    *
650 REM*    POKE 871.RN                         *
660 REM*    POKE 879.PC                         *
670 REM*    POKE 784.AD                         *
680 REM*    SYS(29749)                          *
690 REM*    X=X+XX:Y=Y+YY                       *
700 REM*    NEXT I                              *
710 REM*                                        *
720 REM*    WHERE B$ IS THE CHARACTER STRING TO *
730 REM*    BE PLOTTED. XX IS THE X STEP BETWEEN*
740 REM*    LETTERS FROM LEFT TO RIGHT AND YY IS*
750 REM*    THE Y STEP FROM TOP TO BOTTOM.      *
760 REM******************************************
1000 I=28672:ER=0:CT=0:PRINT""
1010 T=0
1020 READ A
1040 IF A>255 THEN 2000
1050 IF A=-1 THEN 3000
1060 T=T+A:POKE I,A
1070 I=I+1
1080 GOTO 1020
2000 REM
2010 IF T=A THEN 2020
2012 PRINT"LINES"20000+CT"-";
2014 PRINT20000+CT+90;T.A:ER=1:GOTO 2030
2020 PRINT"LINES"20000+CT"-"20000+CT+90"O.K."
2030 CT=CT+100
2040 GOTO 1010
3000 IF ER=1THENEND
3010 POKE51.0:POKE52.112
3020 POKE55.0:POKE56.112:CLR:NEW
20000 DATA72.138.72.152.72.32.32
20010 DATA112.32.71.112.173.17.208
20020 DATA9.32.141.17.208.173.24
20030 DATA208.9.8.141.24.208.104
20040 DATA168.104.170.104.160.0.169
20050 DATA64.133.87.169.63.133.88
20060 DATA169.0.145.87.165.87.240
20070 DATA5.198.87.76.42.112.198
```

36

```
20080 DATA88,165,88,201,31,240,7
20090 DATA169,255,133,87,76,42,112
20095 DATA7518
20100 DATA96,160,0,169,231,133,87
20110 DATA169,7,133,88,165,89,145
20120 DATA87,165,87,240,5,198,87
20130 DATA76,81,112,198,88,165,88
20140 DATA201,3,240,7,169,255,133
20150 DATA87,76,81,112,96,72,152
20160 DATA72,138,72,160,0,24,152
20170 DATA133,95,165,90,201,1,144
20180 DATA12,201,1,208,6,165,89
20190 DATA201,64,144,2,230,95,165
20195 DATA8063
20200 DATA91,201,200,144,2,230,95
20210 DATA165,95,240,6,104,170,104
20220 DATA168,104,96,165,89,41,7
20230 DATA133,94,169,7,56,229,94
20240 DATA133,94,240,9,168,169,1
20250 DATA10,136,208,252,240,2,169
20260 DATA1,133,94,165,91,41,7
20270 DATA133,93,165,91,74,74,74
20280 DATA133,95,133,92,141,109,3
20290 DATA169,0,133,91,133,96,141
20295 DATA7835
20300 DATA110,3,160,6,24,6,95
20310 DATA176,7,6,96,136,208,246
20320 DATA240,6,6,96,230,96,208
20330 DATA245,24,165,91,101,95,133
20340 DATA91,165,92,101,96,133,92
20350 DATA160,3,24,70,90,176,7
20360 DATA70,89,136,208,246,240,10
20370 DATA70,89,165,89,9,128,133
20380 DATA89,208,241,165,89,141,108
20390 DATA3,160,3,24,6,89,176
20395 DATA7499
20400 DATA7,6,90,136,208,246,240
20410 DATA6,6,90,230,90,208,245
20420 DATA169,0,133,87,133,88,165
20430 DATA93,24,101,91,133,87,165
20440 DATA88,101,92,133,88,165,87
20450 DATA24,101,89,133,87,165,88
20460 DATA101,90,24,105,32,133,88
20470 DATA24,160,0,173,16,3,240
20480 DATA15,169,255,69,94,133,94
20490 DATA177,87,37,94,145,87,56
20495 DATA7419
20500 DATA176,6,177,87,5,94,145
20510 DATA87,173,108,3,133,89,169
20520 DATA0,133,90,133,91,133,92
20530 DATA24,14,109,3,14,109,3
```

```
20540 DATA14,109,3,173,109,3,133
20550 DATA91,24,14,109,3,46,110
20560 DATA3,24,14,109,3,46,110
20570 DATA3,173,109,3,24,101,91
20580 DATA133,91,173,110,3,101,92
20590 DATA133,92,24,169,0,133,87
20595 DATA5495
20600 DATA133,88,165,89,24,101,91
20610 DATA133,87,165,92,101,88,24
20620 DATA105,4,133,88,173,111,3
20630 DATA10,10,10,10,141,112,3
20640 DATA160,0,177,87,41,15,13
20650 DATA112,3,145,87,169,1,141
20660 DATA17,3,169,0,133,95,104
20670 DATA170,104,168,104,96,189,0
20680 DATA2,201,32,208,1,232,189
20690 DATA0,2,240,72,138,72,152
20695 DATA6338
20700 DATA72,173,64,3,56,237,60
20710 DATA3,141,68,3,173,65,3
20720 DATA237,61,3,141,69,3,173
20730 DATA66,3,56,237,62,3,141
20740 DATA70,3,173,63,3,237,67
20750 DATA3,141,71,3,169,1,141
20760 DATA94,3,141,96,3,169,0
20770 DATA141,95,3,141,97,3,173
20780 DATA71,3,41,128,240,8,169
20790 DATA255,141,94,3,141,95,3
20795 DATA6072
20800 DATA173,69,3,41,128,240,8
20810 DATA169,255,141,96,3,141,97
20820 DATA3,173,69,3,41,128,240
20830 DATA30,173,69,3,73,255,141
20840 DATA73,3,24,173,68,3,73
20850 DATA255,105,1,141,72,3,173
20860 DATA73,3,105,0,141,73,3
20870 DATA76,112,114,173,68,3,141
20880 DATA72,3,173,69,3,141,73
20890 DATA3,173,71,3,41,128,240
20895 DATA6410
20900 DATA30,173,71,3,73,255,141
20910 DATA75,3,24,173,70,3,73
20920 DATA255,105,1,141,74,3,173
20930 DATA75,3,105,0,141,75,3
20940 DATA76,161,114,173,70,3,141
20950 DATA74,3,173,71,3,141,75
20960 DATA3,173,72,3,56,237,74
20970 DATA3,141,88,3,173,73,3
20980 DATA237,75,3,141,89,3,41
20990 DATA128,240,60,169,255,141,90
20995 DATA6373
```

```
21000 DATA3.141.91.3.169.0.141
21010 DATA92.3.141.93.3.173.74
21020 DATA3.141.76.3.173.75.3
21030 DATA141.77.3.173.72.3.141
21040 DATA78.3.173.73.3.141.79
21050 DATA3.173.71.3.41.128.208
21060 DATA70.169.1.141.90.3.169
21070 DATA0.141.91.3.76.45.115
21080 DATA169.0.141.90.3.141.91
21090 DATA3.169.255.141.92.3.141
21095 DATA5941
21100 DATA93.3.173.72.3.141.76
21110 DATA3.173.73.3.141.77.3
21120 DATA173.74.3.141.78.3.173
21130 DATA75.3.141.79.3.173.69
21140 DATA3.41.128.208.10.169.1
21150 DATA141.92.3.169.0.141.93
21160 DATA3.173.76.3.141.82.3
21170 DATA173.77.3.141.83.3.173
21180 DATA78.3.141.80.3.173.79
21190 DATA3.141.81.3.173.76.3
21195 DATA5592
21200 DATA56.237.78.3.141.84.3
21210 DATA173.77.3.237.79.3.141
21220 DATA85.3.78.77.3.110.76
21230 DATA3.173.78.3.56.237.76
21240 DATA3.141.86.3.173.79.3
21250 DATA237.77.3.141.87.3.173
21260 DATA18.3.141.16.3.173.60
21270 DATA3.133.89.173.61.3.133
21280 DATA90.173.62.3.133.91.32
21290 DATA110.112.173.87.3.41.128
21295 DATA5829
21300 DATA240.60.173.86.3.24.109
21310 DATA80.3.141.86.3.173.87
21320 DATA3.109.81.3.141.87.3
21330 DATA173.60.3.24.109.92.3
21340 DATA141.60.3.173.61.3.109
21350 DATA93.3.141.61.3.173.62
21360 DATA3.24.109.90.3.141.62
21370 DATA3.173.63.3.109.91.3
21380 DATA141.63.3.76.5.116.173
21390 DATA86.3.56.237.84.3.141
21395 DATA5309
21296 DATA86.3.173.87.3.237.85
21410 DATA3.141.87.3.173.60.3
21420 DATA24.109.96.3.141.60.3
21430 DATA173.61.3.109.97.3.141
21440 DATA61.3.173.62.3.24.109
21450 DATA94.3.141.62.3.173.63
21460 DATA3.109.95.3.141.63.3
```

```
21470 DATA173,82,3,56,233,1,141
21480 DATA82,3,173,83,3,233,0
21490 DATA141,83,3,173,17,3,240
21495 DATA5484
21500 DATA4,165,95,208,16,173,83
21510 DATA3,240,3,76,119,115,173
21520 DATA82,3,240,3,76,119,115
21530 DATA104,168,104,170,104,96,72
21540 DATA152,72,138,72,173,103,3
21550 DATA240,5,169,255,141,103,3
21560 DATA173,102,3,201,31,176,3
21570 DATA76,18,117,201,64,144,14
21580 DATA24,201,96;144,3,76,18
21590 DATA117,56,233,64,141,102,3
21595 DATA7159
21600 DATA169,208,133,252,169,0,133
21610 DATA251,173,102,3,160,3,24
21620 DATA10,141,102,3,165,252,105
21630 DATA0,133,252,173,102,3,136
21640 DATA208,239,133,251,169,8,141
21650 DATA106,3,173,14,220,41,254
21660 DATA141,14,220,165,1,41,251
21670 DATA133,1,160,0,177,251,141
21680 DATA102,3,165,1,9,4,133
21690 DATA1,173,14,220,9,1,141
21695 DATA7989
21700 DATA14,220,173,103,3,240,6
21710 DATA77,102,3,141,102,3,173
21720 DATA99,3,141,105,3,173,96
21730 DATA3,141,104,3,169,128,141
21740 DATA107,3,173,102,3,45,107
21750 DATA3,240,18,173,104,3,133
21760 DATA89,173,105,3,133,90,173
21770 DATA100,3,133,91,32,110,112
21780 DATA78,107,3,240,20,173,104
21790 DATA3,24,105,1,141,104,3
21795 DATA6338
21800 DATA173,105,3,105,0,141,105
21810 DATA3,76,196,116,165,251,24
21820 DATA105,1,133,251,165,252,105
21830 DATA0,133,252,24,238,100,3
21840 DATA176,8,206,106,3,240,3
21850 DATA76,133,116,104,170,104,168
21860 DATA104,96
21870 DATA5038,-1
READY.
```

40

HIGH RESOLUTION DEMO

The hi-res loader on pp. 35-40 should be loaded and run once before using this demo.

DESCRIPTION

This demonstration program plots a series of graphs using the line plot and point plot routines. The first graph is of SIN(X), drawn with the axis drawn from co-ordinates 0,100 across the screen. When the graph plot has been completed, the graph is then plotted four times the resolution using the point plot routine.

The next two are 3D graphs, each take about 7 minutes to plot. The reason for the long time in plotting is the extremely complicated calculation needed to transform the three dimensional figure into a two dimensional display on the screen. With all of these graphs, the character plot routine is used to label the graphs.

The variables used are of a standard convention: – X and Y are x and y co-ordinates, X1, Y1 and X2, Y2 are co-ordinates for each end of a line to be plotted, PC is the point colour (0-15), SC is the screen colour (0-15), AD is add or delete (0=add, 1=delete), RN is a reverse or normal flag for use in character plotting (0=normal, 1=reverse), and XX and YY are the x and y steps between characters in the character plot routine.

RUNNING THE PROGRAM

Before this demo is loaded and run, make sure that the hi-res loader has been loaded and run. The loader need only be run once and it will be there until the machine is switched off.

Now the loader is 'resident', the demonstration program may be loaded and run. There are no input parameters. When all three graphs have been plotted, the program will repeat itself. To stop the program hit RUN-STOP/RESTORE.

PROGRAM STRUCTURE

A brief run down of the program lines is as follows:

Lines 10-170	: Rem's to explain what each of the subroutines does
180-200	: Display the text 'Y=SIN(X)' in the bottom corner of the screen.
210-250	: Loop to plot lines from co-ordinates 0,100 to points on a sine curve.
260-290	: Loop to plot points on the same curve but of a smaller step in x.
300	: Pause between graphs.
310	: Function for the first 3D graph.

315	:	Loop for both graphs.
340-350	:	Display the text in the bottom left corner.
360-370	:	Loop to plot 3D graph on the screen.
480	:	Pause between graphs.
490	:	A new equation for a different 3D graph.
500	:	Go back and plot new graph.
510	:	Repeat program.
10000-10030	:	Subroutine to call the set up routine.
11000-11020	:	Subroutine to call the clear hi-res screen routine
12000-12030	:	Subroutine to call the global colour change
13000-13070	:	Subroutine to call the point plot routine
14000-14130	:	Subroutine to call the line plot routine
15000-15150	:	Subroutine to loop and call the character plot routine.

```
10 REM *****************************************
20 REM * THIS PROGRAM DEMONSTRATES THE USE OF  *
30 REM * THE HI-RES GRAPHICS ROUTINES.         *
40 REM *.FOR EASE OF USE. THE POKE AND SYS     *
50 REM * ROUTINES ARE IN SUBROUTINES AT THE END *
60 REM * OF THE PROGRAM THUS.-                  *
70 REM *   GOSUB 10000    (SET UP HIRES SCREEN) *
80 REM *   GOSUB 11000    (CLEAR THE BIT MAP    *
90 REM *                   SCREEN)              *
100 REM*   GOSUB 12000    (DO A GLOBAL COLOUR   *
110 REM*                   CHANGE)              *
120 REM*   GOSUB 13000    (PLOT A POINT)        *
130 REM*   GOSUB 14000    (DRAW A LINE)         *
140 REM*   GOSUB 15000    (DISPLAY A STRING ON THE*
150 REM*                   SCREEN)              *
160 REM*                                        *
170 REM*****************************************
174 REM
175 REM PLOT LINES FROM ORIGIN TO
176 REM POINT ON SINE GRAPH
177 REM
180 PC=1:SC=2:GOSUB 10000
190 B$="Y=SIN(X)":RN=1:AD=0:PC=5
200 X=0:XX=0:Y=192:YY=0:GOSUB 15000
210 PC=1:X1=0:Y1=100:AD=0
220 FOR X2=1 TO 320 STEP 2
230 Y2=100-(90*SIN(X2*π/45.5))
240 GOSUB 14000
250 NEXT X2
254 REM
255 REM PLOT THE SINE CURVE
256 REM
260 FOR X=0 TO 319 STEP .5
270 Y=100-(90*SIN(X*π/45.5))
280 GOSUB 13000
290 NEXT X
300 FOR I=1 TO 1000:NEXT I
310 DEF FNA(Y)=38*(SIN(Y/24)+.48*SIN(3*Y/24))+20
311 REM
312 REM PLOT A 3D GRAPH USING THE FUNCTION
313 REM DEFINED IN LINE 310
314 REM
315 FOR J=1TO2
320 PC=6:SC=12:GOSUB 12000
330 GOSUB 11000
340 B$="3D GRAPH":RN=1:AD=0:PC=0
350 X=0:XX=8:Y=192:YY=0:GOSUB 15000
360 PC=6
370 FOR XX=-100 TO 0 STEP 1
380 K=6:L=0:P=1:Z1=0:M=1
390 Y1=K*INT(SQR(10000-XX*XX)/K)
```

43

```
400 FOR Z=Y1 TO -Y1 STEP -K
410 Y=INT(80+FNA(SQR(XX*XX+Z*Z))-.707106*Z)
420 IF Y<L THEN 470
430 L=Y:Y=190-Y
440 X=M*XX+160
450 GOSUB 13000
460 M=-M:IF M=-1THEN440
470 NEXT Z,XX
480 FOR I=1 TO 1000:NEXT I
484 REM
485 REM A DIFFERENT 3D GRAPH
486 REM
490 DEF FNA(Y)=90*EXP(-Y*Y/600)
500 NEXT J
510 RUN:REM REPEAT
10000 REM SET UP HIRES SCREEN
10010 POKE 89,SC+PC*16
10020 SYS(28672)
10030 RETURN
11000 REM CLEAR GRAPHICS SCREEN
11010 SYS(28704)
11020 RETURN
12000 REM GLOBAL COLOUR CHANGE
12010 POKE 89,SC+PC*16
12020 SYS(28743)
12030 RETURN
13000 REM PLOT A POINT AT X,Y
13010 POKE 89,X-INT(X/256)*256
13020 POKE 90,INT(X/256)
13030 POKE 91,INT(Y)
13040 POKE 784,AD
13050 POKE 879,PC
13060 SYS(28782)
13070 RETURN
14000 REM DRAW A LINE BETWEEN X1,Y1
14010 REM AND X2,Y2
14020 POKE 828,X1-INT(X1/256)*256
14030 POKE 829,INT(X1/256)
14040 POKE 830,Y1-INT(Y1/256)*256
14050 POKE 831,INT(Y1/256)
14060 POKE 832,X2-INT(X2/256)*256
14070 POKE 833,INT(X2/256)
14080 POKE 834,Y2-INT(Y2/256)*256
14090 POKE 835,INT(Y2/256)
14100 POKE 786,AD
14110 POKE 879,PC
14120 SYS(29158)
14130 RETURN
15000 REM PLOT CHARACTER A$ ON TO THE
15010 REM SCREEN WITH TOP LEFT COORD-
```

```
15020 REM INATES X,Y
15030 FORI=1TOLEN(B$)
15040 A$=MID$(B$,I,1)
15050 POKE 866,X-INT(X/256)*256
15060 POKE 867,INT(X/256)
15070 POKE 868,Y:POKE689,0
15080 POKE 870,ASC(A$)
15090 POKE 871,RN
15100 POKE 784,AD
15110 POKE 879,PC
15120 SYS(29749)
15130 X=X+XX:Y=Y+YY
15140 NEXT
15150 RETURN
READY.
```

3D DRAWING 1

The hi-res loader on pp. 35-40 should be loaded and run once before using this demo.

DESCRIPTION

3 Dimensional shapes may be displayed in two dimensions by having the optical effect of being 3 dimensional. This is achieved by using a transformation matrix. This simply requires the addition of an extra axis – the Z axis – to the X and Y axis used in a two dimensional transformation matrix (see program MOVE). The transformation matrix consists of sixteen equations, they are stored in lines 3010 to 3160. The mathematics are rather awkward to explain, however, for those interested, the book 'Principles of Interactive Graphics' by Newman and Sproul can be recommended.

RUNNING THE PROGRAM

There are no input parameter values since they are all within the program as LET statements. There are nine parameter values which control the movement, rotation of scaling of the shape, these are set in lines 120-200. Lines 120 and 140 contain the X, Y, and Z scaling factors–full size = 1, half size = 0.5 etc. The rotational angles of the shape in either one of the three axes are stored in lines 180-200; note that since these angles must be in radians they are multiplied by Pi. The movement of the shape in the X, Y, and Z axes is stored in lines 150-170, and is the number of pixels in either direction from the original co-ordinates stored in the shape table.
The object shape is stored in a shape table consisting of two parts, the first is simply of the X, Y, and Z co-ordinates, of each corner co-ordinate comprising the shape. The second part is a table of connections of pairs of points between which a line should be drawn. The number of edges in the shape is stored as the variable 'NE' and the number of co-ordinate points between which the edges are connected is stored as variable 'NP'. The co-ordinate table is stored as data statements in lines 1210-1220, and the connection table in lines 1310-1330.

PROGRAM STRUCTURE

A brief run down of the program lines is as follows:
Line 70   : Draw border around screen using subroutine at 900.

100-110	: Set up transform matrix arrays.
120-140	: X, Y, and Z scaling factors.
150-170	: X, Y, and Z axis movement of shape from initial position.
180-200	: Angle of X, Y, and Z axis rotation in radians.
410-530	: Main program execution routine.
900-960	: Border drawing subroutine.
1000-1170	: Load shape data into arrays – array S contains the co-ordinate table of the original shape – array E contains the line connection data – array M contains the transformed co-ordinate data.
1200-1220	: Data statements containing co-ordinate shape data as X, Y, and Z for each corner point; note that the first three values comprise the co-ordinates for point 1, the second three for point 2 etc.
2000-2240	: Draw the shape.
3000-3160	: Perform transformation matrix calculations.
3200-3350	: Set up scaling and transformation matrix.
4000-4080	: Perform the transformation on each co-ordinate point within the shape table.
5000-5090	: Find centre of shape.
10000-10030	: Subroutine to call the set up routine.
11000-11090	: Subroutine to call the line plot routine.

47

```
1 REM 3D DRAWING 1
2 REM *************************
3 REM
10 REM A THREE DIMENSIONAL SHAPE IS DRAWN BY THIS PROGRAM
20 REM THE ROTATION POSITION AND SCALE OF THE OBJECT
30 REM CAN BE CHANGED TO GIVE DIFFERENT VIEWING ANGLES.
35 REM .
40 REM SET COLOURS
50 PC=6:SC=12
60 GOSUB 10000
65 REM
70 REM DRAW BORDER AROUND SCREEN
75 REM
80 GOSUB 900
85 REM
90 REM SET UP CONSTANTS, VARIABLES, AND ARRAYS
95 REM
100 DIM A(4,4)
110 DIM B(4,4)
120 SX=.3
130 SY=.3
140 SZ=.3
150 TX=1
160 TY=1
170 TZ=1
180 RX=40*π/180
190 RY=20*π/180
200 RZ=50*π/180
400 REM MAIN PROGRAM LOOP
410 GOSUB 1000
420 GOSUB 5000
430 GOSUB 3000
440 GOSUB 4000
450 GOSUB 2000
500 GET A$:IF A$="" THEN 500
510 POKE53280,14:POKE53281,6:PRINT"◻◻";
520 POKE53265,27:POKE53272,21
530 END
900 REM BORDER DRAWING SUBROUTINE
905 REM
910 X1=0:Y1=0
920 X2=319:Y2=0:AD=0:GOSUB 11000
930 X1=X2:Y1=Y2:Y2=199:GOSUB 11000
940 X1=X2:Y1=Y2:X2=0:GOSUB 11000
950 X1=X2:Y1=Y2:Y2=0:GOSUB 11000
960 RETURN
995 REM
1000 REM INITIALISE SHAPE
1005 REM
1010 NP=8
1020 NE=12
```

48

```
1030 REM
1040 DIM S(3,NP)
1050 DIM E(NE,2)
1060 DIM M(3,NP)
1100 REM
1110 FOR N=1 TO NP
1120 READ S(1,N),S(2,N),S(3,N)
1130 NEXT N
1140 FOR K=1 TO NE
1150 READ E(K,1),E(K,2)
1170 NEXT K
1195 REM
1200 REM X,Y,Z POINT COORDINATES
1210 DATA 50,0,200,250,0,200,250,0,0,50,0,0
1220 DATA 50,200,200,250,200,200,250,200,0,50,200,0
1295 REM
1300 REM CONNECTION DATA
1305 REM
1310 DATA 1,2,2,3,3,4,4,1
1320 DATA 5,1,2,6,4,8,7,3
1330 DATA 6,5,5,8,8,7,7,6
1900 RETURN
1995 REM
2000 REM DRAW SHAPE
2005 REM
2020 FOR K=1 TO NE
2030 V1=E(K,1)
2040 V2=E(K,2)
2045 IF V1=0 THEN 2240
2050 X1=M(1,V1)
2060 Y1=M(2,V1)
2070 X2=M(1,V2)
2080 Y2=M(2,V2)
2220 AD=0:PC=2:GOSUB 11000
2240 NEXT K
2900 RETURN
2995 REM
3000 REM SET TRANSFORMATION MATRIX
3005 REM
3010 A(1,1)=COS(RY)*COS(RZ)
3020 A(1,2)=COS(RY)*SIN(RZ)
3030 A(1,3)=-SIN(RY)
3040 A(1,4)=0
3050 A(2,1)=COS(RX)*(-SIN(RZ))+SIN(RX)*SIN(RY)*COS(RZ)
3060 A(2,2)=COS(RX)*COS(RZ)+SIN(RX)*SIN(RY)*SIN(RZ)
3070 A(2,3)=SIN(RX)*COS(RY)
3080 A(2,4)=0
3090 A(3,1)=(-SIN(RX))*(-SIN(RZ))+COS(RX)*SIN(RY)*COS(RZ)
3100 A(3,2)=-SIN(RX)*COS(RZ)+COS(RZ)*SIN(RY)*SIN(RZ)
3110 A(3,3)=COS(RX)*COS(RY)
```

```
3120 A(3,4)=0
3130 A(4,1)=0
3140 A(4,2)=0
3150 A(4,3)=0
3160 A(4,4)=1
3195 REM
3200 REM SET UP SCALING AND TRANSLATION MATRIX
3205 REM
3210 B(1,1)=SX*A(1,1)
3220 B(1,2)=SX*A(1,2)
3230 B(1,3)=SX*A(1,3)
3240 REM
3250 B(2,1)=SY*A(2,1)
3260 B(2,2)=SY*A(2,2)
3270 B(2,3)=SY*A(2,3)
3280 REM
3290 B(3,1)=SZ*A(3,1)
3300 B(3,2)=SZ*A(3,2)
3310 B(3,3)=SZ*A(3,3)
3320 REM
3330 B(4,1)=TX
3340 B(4,2)=TY
3350 B(4,3)=TZ
3900 RETURN
3995 REM
4000 REM PERFORM TRANSLATION
4005 REM
4010 FOR Q=1 TO NP
4015 REM
4020 XT=S(1,Q)-XC
4030 YT=S(2,Q)-YC
4040 ZT=S(3,Q)-ZC
4045 REM
4050 M(1,Q)=XC+(XT*B(1,1)+YT*B(2,1)+ZT*B(3,1)+B(4,1))
4060 M(2,Q)=YC+(XT*B(1,2)+YT*B(2,2)+ZT*B(3,2)+B(4,2))
4070 M(3,Q)=ZC+(XT*B(1,3)+YT*B(2,3)+ZT*B(3,3)+B(4,3))
4080 NEXT Q
4900 RETURN
4995 REM
5000 REM FIND CENTROID
5005 REM
5010 P=0:Q=0:R=0
5020 FOR I=1 TO NP
5030 P=P+S(1,I)
5040 Q=Q+S(2,I)
5050 R=R+S(3,I)
5060 NEXT I
5070 XC=P/NP
5080 YC=Q/NP
5090 ZC=R/NP
```

50

```
5900 RETURN
10000 REM SET UP HIRES SCREEN
10010 POKE89.SC+PC*16
10020 SYS(28672):POKE53280.SC
10030 RETURN
11000 REM DRAW A LINE BETWEEN X1.Y1 AND X2.Y2
11010 POKE828.X1-INT(X1/256)*256
11020 POKE829.INT(X1/256)
11030 POKE830.Y1:POKE831.0
11040 POKE832.X2-INT(X2/256)*256
11050 POKE833.INT(X2/256)
11060 POKE834.Y2:POKE835.0
11070 POKE786.AD:POKE879.PC
11080 SYS(29158)
11090 RETURN
READY.
```

3D DRAWING 2

The hi-res loader on pp. 35-40 should be loaded and run once before using this demo.

DESCRIPTION

This program is identical to the program 3D DRAWING 1 except that an additional subroutine has been added to remove hidden lines. Hidden lines are those lines which lie out of sight of the viewer and are hidden behind the front surfaces. By removing these hidden lines the shape of the object becomes much clearer. The subroutine which checks for hidden lines is located between line numbers 6000 and 6150.

RUNNING THE PROGRAM

The parameters and data tables required for this program are the same as those used for the program 3D DRAWING 1, consult this program for information. Note that the connection table now describes object faces rather than lines.

PROGRAM STRUCTURE

A brief run down of the program lines is as follows:
Lines 1-5995 are identical to 3D DRAWING 1 except the following lines:

115-117	: The addition of two more arrays.
450	: Is now GOSUB 6000.
1010-1170	: Slight alterations required.
1310-1360	: Different data.
6000-6140	: Subroutine to check for hidden surfaces.

```
1 REM 3D DRAWING 2
2 REM ***********************
3 REM
10 REM A THREE DIMENSIONAL SHAPE IS DRAWN BY THIS PROGRAM
20 REM THE ROTATION POSITION AND SCALE OF THE OBJECT
30 REM CAN BE CHANGED TO GIVE DIFFERENT VIEWING ANGLES.
35 REM HIDDEN LINES ARE NOT DRAWN
36 REM
40 REM SET COLOURS
50 PC=6:SC=12
60 GOSUB 10000
65 REM
70 REM DRAW BORDER AROUND SCREEN
75 REM
80 GOSUB 900
85 REM
90 REM SET UP CONSTANTS, VARIABLES, AND ARRAYS
95 REM
100 DIM A(4,4)
110 DIM B(4,4)
115 DIM C(3)
117 DIM D(3)
120 SX=.3
130 SY=.3
140 SZ=.3
150 TX=1
160 TY=1
170 TZ=1
180 RX=40*π/180
190 RY=20*π/180
200 RZ=50*π/180
400 REM MAIN PROGRAM LOOP
410 GOSUB 1000
420 GOSUB 5000
430 GOSUB 3000
440 GOSUB 4000
450 GOSUB 6000
500 GET A$:IF A$="" THEN 500
510 POKE53280,14:POKE53281,6:PRINT"◻";
520 POKE53265,27:POKE53272,21
530 END
900 REM BORDER DRAWING SUBROUTINE
905 REM
910 X1=0:Y1=0
920 X2=319:Y2=0:AD=0:GOSUB 11000
930 X1=X2:Y1=Y2:Y2=199:GOSUB 11000
940 X1=X2:Y1=Y2:X2=0:GOSUB 11000
950 X1=X2:Y1=Y2:Y2=0:GOSUB 11000
960 RETURN
995 REM
```

53

```
1000 REM INITIALISE SHAPE
1005 REM
1010 NP=8
1020 NE=4
1030 NF=6
1040 DIM S(3,NP)
1050 DIM E(NF,NE,2)
1060 DIM M(3,NP)
1100 REM
1110 FOR N=1 TO NP
1120 READ S(1,N),S(2,N),S(3,N)
1130 NEXT N
1135 FOR F=1 TO NF
1140 FOR K=1 TO NE
1150 READ E(F,K,1),E(F,K,2)
1170 NEXT K,F
1195 REM
1200 REM X,Y,Z POINT COORDINATES
1210 DATA 50,0,200,250,0,200,250,0,50,0,0
1220 DATA 50,200,200,250,200,200,250,200,0,50,200,0
1295 REM
1300 REM CONNECTION DATA
1305 REM
1310 DATA 1,2,2,3,3,4,4,1
1320 DATA 5,1,1,4,4,8,8,5
1330 DATA 6,5,5,8,8,7,7,6
1340 DATA 2,6,6,7,7,3,3,2
1350 DATA 1,5,5,6,6,2,2,1
1360 DATA 3,7,7,8,8,4,4,3
1900 RETURN
1995 REM
2000 REM DRAW SHAPE
2005 REM
2020 FOR K=1 TO NE
2030 V1=E(F,K,1)
2040 V2=E(F,K,2)
2045 IF V1=0 THEN 2240
2050 X1=M(1,V1)
2060 Y1=M(2,V1)
2070 X2=M(1,V2)
2080 Y2=M(2,V2)
2220 AD=0:PC=2:GOSUB 11000
2240 NEXT K
2900 RETURN
2995 REM
3000 REM SET TRANSFORMATION MATRIX
3005 REM
3010 A(1,1)=COS(RY)*COS(RZ)
3020 A(1,2)=COS(RY)*SIN(RZ)
3030 A(1,3)=-SIN(RY)
3040 A(1,4)=0
```

54

```
3050 A(2,1)=COS(RX)*(-SIN(RZ))+SIN(RX)*SIN(RY)*COS(RZ)
3060 A(2,2)=COS(RX)*COS(RZ)+SIN(RX)*SIN(RY)*SIN(RZ)
3070 A(2,3)=SIN(RX)*COS(RY)
3080 A(2,4)=0
3090 A(3,1)=(-SIN(RX))*(-SIN(RZ))+COS(RX)*SIN(RY)*COS(RZ)
3100 A(3,2)=-SIN(RX)*COS(RZ)+COS(RZ)*SIN(RY)*SIN(RZ)
3110 A(3,3)=COS(RX)*COS(RY)
3120 A(3,4)=0
3130 A(4,1)=0
3140 A(4,2)=0
3150 A(4,3)=0
3160 A(4,4)=1
3195 REM
3200 REM SET UP SCALING AND TRANSLATION MATRIX
3205 REM
3210 B(1,1)=SX*A(1,1)
3220 B(1,2)=SX*A(1,2)
3230 B(1,3)=SX*A(1,3)
3240 REM
3250 B(2,1)=SY*A(2,1)
3260 B(2,2)=SY*A(2,2)
3270 B(2,3)=SY*A(2,3)
3280 REM
3290 B(3,1)=SZ*A(3,1)
3300 B(3,2)=SZ*A(3,2)
3310 B(3,3)=SZ*A(3,3)
3320 REM
3330 B(4,1)=TX
3340 B(4,2)=TY
3350 B(4,3)=TZ
3900 RETURN
3995 REM
4000 REM PERFORM TRANSLATION
4005 REM
4010 FOR Q=1 TO NP
4015 REM
4020 XT=S(1,Q)-XC
4030 YT=S(2,Q)-YC
4040 ZT=S(3,Q)-ZC
4045 REM
4050 M(1,Q)=XC+(XT*B(1,1)+YT*B(2,1)+ZT*B(3,1)+B(4,1))
4060 M(2,Q)=YC+(XT*B(1,2)+YT*B(2,2)+ZT*B(3,2)+B(4,2))
4070 M(3,Q)=ZC+(XT*B(1,3)+YT*B(2,3)+ZT*B(3,3)+B(4,3))
4080 NEXT Q
4900 RETURN
4995 REM
5000 REM FIND CENTROID
5005 REM
5010 P=0:Q=0:R=0
5020 FOR I=1 TO NP
5030 P=P+S(1,I)
```

```
5040 Q=Q+S(2,I)
5050 R=R+S(3,I)
5060 NEXT I
5070 XC=P/NP
5080 YC=Q/NP
5090 ZC=R/NP
5900 RETURN
5995 REM
6000 REM HIDDEN SURFACE CHECK
6005 REM
6010 FOR F=1 TO NF
6020 FOR J=1 TO 3
6030 C(J)=M(J,E(F,1,2))-M(J,E(F,1,1))
6040 D(J)=M(J,E(F,2,1))-M(J,E(F,2,2))
6050 NEXT J
6060 P1=C(2)*D(3)-C(3)*D(2)
6070 P2=C(3)*D(1)-C(1)*D(3)
6080 P3=C(1)*D(2)-C(2)*D(1)
6090 Q1=1-M(1,E(F,1,2))
6100 Q2=1-M(2,E(F,1,2))
6110 Q3=500-M(3,E(F,1,2))
6120 W=P1*Q1+P2*Q2+P3*Q3
6130 IF W>=0 THEN GOSUB 2000
6140 NEXT F
6150 RETURN
10000 REM SET UP HIRES SCREEN
10010 POKE89,SC+PC*16
10020 SYS(28672):POKE53280,SC
10030 RETURN
11000 REM DRAW A LINE BETWEEN X1,Y1 AND X2,Y2
11010 POKE828,X1-INT(X1/256)*256
11020 POKE829,INT(X1/256)
11030 POKE830,Y1:POKE831,0
11040 POKE832,X2-INT(X2/256)*256
11050 POKE833,INT(X2/256)
11060 POKE834,Y2:POKE835,0
11070 POKE786,AD:POKE979,PC
11080 SYS(29158)
11090 RETURN
READY.
```

MOVE

The hi-res loader on pp. 35-40 should be loaded and run once before using this demo.

DESCRIPTION

A transformation matrix can be used to cover all manipulation of a shape for rotation, translation, and scaling. The primary purpose of this program is to show how a shape can be moved about the screen, but it also embodies the capability of scaling and rotation. The transformation matrix consists of six quotations. These equations are stored in lines 3000 – 3100. Equations 1 to 4 consist of the rotation transform equation multiplied by a scaling factor, equations 5 and 6 do the movement by adding an offset to the shape position. The shape can be translated to any part of the screen, rotated through 360 degrees and stretched in either X or Y axis or both. Note that the actual moving is done in the program, what appears on the screen is the end result, and due to the large amount of calculation required, is necessarily rather slow.

RUNNING THE PROGRAM

There are no input parameter values since they are all within the program as 'LET' statements. There are six parameter values which control the movement, rotation, or scaling of the shape, these are set in lines 120 to 160. Lines 120 and 130 contain the X and Y scaling factors – full size = 1, half size = 0.5 etc. The rotational angle of the shape is stored as the variable RZ in line 140, note that since this angle must be in radians it is multiplied by Pi/180. The movement of the shape in the X and Y axes is stored in lines 150 and 160, and is the number of pixels in either direction from the original co-ordinates stored in the shape table
   The object is stored in a shape table. This table consists simply of the X and Y co-ordinates at the end of each line comprising the shape. It should be noted that the number of co-ordinate pairs is one more than the number of lines in the shape. The number of lines in the shape is stored as the variable NP as the first value in the data table. The data table is stored as data statements in lines 1110 to 1130. Try designing your own shapes on graph paper and then entering the new values into the data statements.

PROGRAM STRUCTURE

A brief run down of the program lines is as follows:
line 90   : Draw border around screen using subroutines at

```
1 REM MOVE
2 REM ************************
3 REM
10 REM THIS PROGRAM USES MATRIX TRANSFORMATION TO
20 REM MOVE,ROTATE, OR SCALE A TWO DIMENSIONAL SHAPE
30 REM
40 REM
50 REM SET COLOURS
60 SC=12:PC=6
70 GOSUB 10000
75 REM
80 REM DRAW BORDER
90 GOSUB 400
95 REM
100 REM SET UP CONSTANTS, VARIABLES, AND ARRAYS
105 REM
110 DIM A(3,3)
120 SX=1
130 SY=1
140 RZ=80*π/180
150 TX=-50
160 TY=2
190 REM
200 REM MAIN PROGRAM LOOP
205 REM
210 GOSUB 1000
220 GOSUB 2000
230 GOSUB 3000
240 GOSUB 4000
250 GOSUB 5000
260 GET A$:IF A$="" THEN 260
270 POKE53280,14:POKE53281,6:PRINT"◻◻":
280 POKE53272,21:POKE53265,27
290 END
395 REM
400 REM DRAW BORDER
405 REM
410 X1=0:Y1=0:AD=0
420 X2=319:Y2=0:GOSUB 11000
430 X1=X2:Y1=Y2:Y2=199:GOSUB 11000
440 X1=X2:Y1=Y2:X2=0:GOSUB 11000
450 X1=X2:Y1=Y2:Y2=0:GOSUB 11000
460 RETURN
995 REM
1000 REM INITIALISE SHAPE
1005 REM
1010 READ NP
1020 DIM X(NP+1),Y(NP+1),U(NP+1),V(NP+1)
1030 FOR I=1 TO NP+1
```

```
1040 READ X(I),Y(I)
1050 NEXT I
1090 REM
1100 REM SHAPE DATA
1105 REM
1110 DATA 5
1120 DATA 100,100,150,120,175,75
1130 DATA 150,30,100,50,100,100
1200 RETURN
1995 REM
2000 REM FIND CENTRE OF SHAPE
2005 REM
2010 CX=0:CY=0
2020 FOR C=1 TO NP
2030 CX=CX+X(C)
2040 CY=CY+Y(C)
2050 NEXT C
2060 CX=CX/NP
2070 CY=CY/NP
2080 RETURN
2995 REM
3000 REM SET TRANSFORMATION MATRIX
3005 REM
3010 A(1,1)=SX*COS(RZ)
3020 A(1,2)=SX*SIN(RZ)
3030 REM
3040 A(2,1)=SY*(-SIN(RZ))
3050 A(2,2)=SY*COS(RZ)
3060 REM
3070 A(3,1)=TX
3080 A(3,2)=TY
3090 REM
3100 RETURN
3995 REM
4000 REM DO TRANSFORMATION
4005 REM
4010 FOR Q=1 TO NP+1
4020 XT=X(Q)-CX
4030 YT=Y(Q)-CY
4040 U(Q)=CX+(XT*A(1,1)+YT*A(2,1)+A(3,1))
4050 V(Q)=CY+(XT*A(1,2)+YT*A(2,2)+A(3,2))
4060 NEXT Q
4070 RETURN
4995 REM
5000 REM DRAW SHAPE
5005 REM
5010 FOR Q=1 TO NP
5020 X1=U(Q):Y1=V(Q)
5030 X2=U(Q+1):Y2=V(Q+1)
5190 AD=0:PC=0:GOSUB 11000
5210 NEXT Q
```

```
5220 RETURN
10000 POKE89.SC+PC*16
10010 SYS(28672):POKE53280,SC
10020 RETURN
11000 POKE 828,X1-INT(X1/256)*256
11010 POKE 829,INT(X1/256)
11020 POKE 830,Y1:POKE 831,0
11030 POKE 832,X2-INT(X2/256)*256
11040 POKE 833,INT(X2/256)
11050 POKE 834,Y2:POKE 835,0
11060 POKE 786,AD
11070 POKE 879,PC
11080 SYS(29158)
11090 RETURN
READY.
```

ELLIPSE

The hi-res loader on pp. 35-40 should be loaded and run once before using this demo.

DESCRIPTION

This program allows the user to plot ellipses on the screen (circles may also be plotted by setting the elliptical offsets in the X and the Y axis to the same values). The ellipse is drawn with a variable dot spacing, which is input along with the centre co-ordinates, the radius, and the elliptical offsets. Since characters cannot be displayed directly onto the screen in Hi-Res Mode, the input routines will not display the data entry. This has been overcome by using the character plot routine to display the input characters using the 'GET' command. This routine is used throughout the inputs in the program, but is very limited, there is no facility to delete a character (any character other than 0-9, '.' or '-' will act as a return in a normal input).

RUNNING THE PROGRAM

When run, a border is drawn around the screen plotting area and a '?' will appear at the bottom outside the border. This is a prompt to enter the X and Y centre co-ordinates of the ellipse. When the co-ordinates have been entered, then the radius is asked for, then the dot spacing, and finally the elliptical offsets in X and Y.

PROGRAM STRUCTURE

A brief run down of the program lines is as follows.

Lines 60-70	: Set colours and go into hi-res mode.
90	: Draw border round screen using subroutine at 400.
110-119	: Input centre co-ordinates for the ellipse.
120-126	: Input radius of ellipse.
130-136	: Input dot spacing for ellipse.
140-149	: Input elliptical offsets for ellipse.
210-290	: Draw ellipse.
300	: Repeat for another go.
350-370	: if XC<0 or YC<0 then restore registers and end.
400-460	: Draw a border around display area.
10000-10030	: Subroutine to call the set up routine.
11000-11100	: Subroutine to call the line plot routine.
12000-12060	: Subroutine to call the point plot routine.
13000-13090	: Subroutine to call the character plot routine.

```
1 REM ELLIPSE
2 REM ************************
3 REM
10 REM ROUTINE TO DRAW AN ELLIPSE USIN OFFSETS
20 REM SPACING BETWEEN THE DOTS USED
30 REM TO DRAW THE ELLIPSE IS VARIABLE
40 REM
50 REM SET COLOURS
55 REM
60 SC=12:PC=6
70 GOSUB 10000
75 REM
80 REM DRAW BORDER
85 REM
90 GOSUB 400
95 REM
100 REM INPUT ELLIPSE DRAWING PARAMETERS
110 REM COORDINATES OF ELLIPSE CENTRE
111 Z$="":T=16:TT=192:PC=0:RN=0:AD=0
112 RN=0:A$="?":GOSUB 13000
113 T=40:GOSUB 500
114 XC=Z:Z$=""
115 A$=".":GOSUB 13000:T=T+8
116 GOSUB 500
117 YC=Z
118 IF XC<0 OR YC<0 THEN 350
119 FOR T=0 TO 160 STEPB:AD=1:RN=1:A$=" ":GOSUB 13000:NEXT T
120 REM ELLIPSE RADIUS
121 Z$="":T=16:AD=0:RN=0
122 A$="?":GOSUB 13000
123 T=40:GOSUB 500
124 RA=Z
126 FOR T=0 TO 160 STEP 8:RN=1:AD=1:A$=" ":GOSUB 13000:NEXT
130 REM DOT SPACING
131 Z$="":T=16:AD=0:RN=0
132 A$="?":GOSUB 13000
133 T=40:GOSUB 500
134 DS=Z
136 FOR T=0 TO 160 STEP 8:RN=1:AD=1:A$=" ":GOSUB 13000:NEXT
140 REM ELLIPTICAL OFFSETS IN X AND Y AXIS
141 Z$="":T=16:RN=0:AD=0
142 A$="?":GOSUB 13000
143 T=40:GOSUB 500
144 OX=Z:Z$=""
145 A$=".":GOSUB 13000:T=T+B
146 GOSUB 500
147 OY=Z
149 FOR T=0 TO 160 STEP B:RN=1:AD=1:A$=" ":GOSUB 13000:NEXT
195 REM
200 REM DRAW ELLIPSE
```

```
205 REM
210 DS=DS*π/180:RN=0:AD=0
220 R=RA
230 FOR P=0 TO 2*π STEP DS
235 REM
240 X=R*COS(P)*OX
250 Y=R*SIN(P)*OY
270 X=X+XC:Y=Y+YC
275 IF X<0 OR Y<0 OR Y>190 THEN 290
280 PC=2:GOSUB 12000
290 NEXT P
300 GOTO 100
350 POKE53265,27:POKE53272,21
360 POKE53280,14:POKE53281,6:PRINT"◼◼";
370 END
395 REM
400 REM BORDER DRAWING ROUTINE
405 REM
410 X1=0:Y1=0:AD=0
420 X2=319:Y2=0:GOSUB 11000
430 X1=X2:Y1=Y2:Y2=190:GOSUB 11000
440 X1=X2:Y1=Y2:X2=0:GOSUB 11000
450 X1=X2:Y1=Y2:Y2=0:GOSUB 11000
460 RETURN
495 REM
500 REM INPUT DATA
505 REM
510 GET A$:IF A$="" THEN 510
520 IF (ASC(A$)<48 OR ASC(A$)>57) AND A$<>"-" AND A$<>"."
THEN 550
530 RN=0:GOSUB 13000:T=T+8
540 Z$=Z$+A$:GOTO 510
550 Z=VAL(Z$)
560 RETURN
10000 REM SET UP GRAPHICS SCREEN
10010 POKE89,SC+PC*16
10020 SYS(28672):POKE53280,SC
10030 RETURN
11000 REM DRAW A LINE BETWEEN X1,Y1 AND X2,Y2
11010 POKE828,X1-INT(X1/256)*256
11020 POKE829,INT(X1/256)
11030 POKE830,Y1:POKE831,0
11040 POKE832,X2-INT(X2/256)*256
11050 POKE833,INT(X2/256)
11060 POKE834,Y2:POKE835,0
11070 POKE786,AD
11080 POKE879,PC
11090 SYS(29158)
11100 RETURN
12000 REM PLOT A POINT
12010 POKE89,X-INT(X/256)*256
12020 POKE90,INT(X/256)
```

64

```
12030 POKE91,Y
12040 POKE784,AD
12045 POKE879,PC
12050 SYS(28782)
12060 RETURN
13000 REM DISPLAY A CHARACTER
13010 POKE866,T-INT(T/256)*256
13020 POKE867,INT(T/256)
13030 POKE868,TT:POKE869,0
13040 POKE870,ASC(A$)
13050 POKE871,RN
13060 POKE784,AD
13070 POKE879,PC
13080 SYS(29749)
13090 RETURN
READY.
```

INTERPOLATE

DESCRIPTION

Determining a set of data is all very well, but it is the interpolation of that data that produces the all important results. One common method of doing this is to take the data and turn it into points on a graph, and then perform the interpolation on those points. The program 'interpolate' does just that, by assuming that you already have your data in the form of X and Y co-ordinates. You could quite easily incorporate your own data into this program simply by changing the data statements in line 180.

RUNNING THE PROGRAM

The main bulk of the work is done by a) line 180, which stores the data as X, Y co-ordinates, and b) line 200, which determines which point we start at (here it is the first one), which one we finish at (here it is the twelfth), and which points we interpolate between (here it is every one, although by changing the variable SP in line 200 we could easily take every other point, for instance). Once we've calculated the scaling factors in lines 410 to 490, and turned those into point increments in lines 510 and 520, we plot the actual point in line 640, and the line between each point by the routine in lines 670 to 720.

PROGRAM STRUCTURE

A brief description of the program lines is as follows:

Lines		
Lines 60-70	:	Set colours and go into hi-res mode
90	:	Draw border round screen using subroutine at 1000
110-160	:	Read and store data
180	:	Data stored as X, Y co-ordinates
200	:	Determinate start and end points and separation
220-230	:	Determine position and dimensions of graph screen
310-385	:	Draw border round graph and label graph
410-490	:	Determine scaling factor
510-520	:	Convert scaling factors to point increments
610-720	:	Point and line drawing routine
1000-1060	:	Draw border around screen
10000-10020	:	Subroutine to call set up routine
11000-11090	:	Subroutine to call line plot routine
12000-12060	:	Subroutine to call point plot routine
13000-13120	:	Subroutine to call character plot routine.

```
1 REM INTERPOLATE
2 REM ************************
3 REM
10 REM PROGRAM TO DRAW A GRAPH BY INTERPOLATING
20 REM A SET OF POINTS STORED AS DATA STATEMENTS IN
30 REM LINE 180.
45 REM
50 REM SET COLOURS
55 REM
60 SC=12:PC=6
70 GOSUB 10000
75 REM
80 REM DRAW BORDER AROUND SCREEN
85 REM
90 GOSUB 1000
95 REM
100 REM INITIALISE DATA
105 REM
110 DIM X(12)
120 DIM Y(12)
130 FOR I=1 TO 12
140 READ X(I)
150 READ Y(I)
160 NEXT I
165 REM
170 REM DATA STORED AS X AND Y COORDINATE
175 REM
180 DATA1,10,2,25,3,30,4,20,5,40,6,30,7,50,8,20,9,25,10,50,
11,30,12,20
185 REM
190 REM MIN DIMENSION =1, MAX =12, SEPERATION =1
195 REM
200 DN=1:DX=12:SP=1
205 REM
210 REM POSITION AND DIMENSIONS OF GRAPH ON SCREEN
215 REM
220 XL=20:XR=300
230 YB=170:YT=50
295 REM
300 REM DRAW BORDER AROUND GRAPH
305 REM
310 X3=XR+10:Y3=YB+10:AD=0:PC=0
320 X4=XR+10:Y4=YT-10:GOSUB 11000
330 X3=X4:Y3=Y4:X4=XL-10:GOSUB 11000
340 X3=X4:Y3=Y4:Y4=YB+10:GOSUB 11000
350 X3=X4:Y3=Y4:X4=XR+10:GOSUB 11000
355 XI=(XR-XL)/(DX-DN)
360 FOR X=XL TO XR STEP XI
365 FOR A=10 TO 13
370 Y=YB+A:GOSUB 12000
375 NEXT A:NEXT X
```

```
380 A$="INTERPOLATED GRAPH":X=80:Y=20:XX=8:YY=0
385 RN=0:AD=0:PC=2:GOSUB 13000
395 REM
400 REM CALCULATE SCALING FACTORS
405 REM
410 Y1=-1000000
420 Y2=1000000
430 X1=Y1:X2=Y2
440 FOR I=DN TO DX STEP SP
450 IF Y1<Y(I) THEN Y1=Y(I)
460 IF Y2>Y(I) THEN Y2=Y(I)
470 IF X1<X(I) THEN X1=X(I)
480 IF X2>X(I) THEN X2=X(I)
490 NEXT I
495 REM
500 REM CONVERT SCALING FACTORS INTO POINT INCREMENTS
505 REM
510 A=(XR-XL)/(X1-X2)
520 B=(YB-YT)/(Y1-Y2)
595 REM
600 REM PLOT GRAPH
605 REM
610 FOR I=DN TO DX STEP SP
620 X=(XL+(X(I)-X2)*A):X3=X
630 Y=(YB-(Y(I)-Y2)*B):Y3=Y
660 PC=1:AD=0:GOSUB 12000
670 Q=I+SP
680 IF Q>DX THEN 900
690 X=(XL+(X(Q)-X2)*A):X4=X
700 Y=(YB-(Y(Q)-Y2)*B):Y4=Y
710 GOSUB 11000
720 NEXT I
900 GETA$:IFA$=""THEN900
910 POKE53280,14:POKE53281,6:PRINT"{CLR}":
920 POKE53265,27:POKE53272,21
930 END
1000 REM DRAW BORDER AROUND SCREEN
1005 REM
1010 X3=0:Y3=0:AD=0
1020 X4=319:Y4=0:GOSUB 11000
1030 X3=X4:Y3=Y4:Y4=199:GOSUB 11000
1040 X3=X4:Y3=Y4:X4=0:GOSUB 11000
1050 X3=X4:Y3=Y4:Y4=0:GOSUB 11000
1060 RETURN
10000 POKE89,SC+PC*16
10010 SYS(28672):POKE53280,SC
10020 RETURN
11000 POKE 828,X3-INT(X3/256)*256
11010 POKE829,INT(X3/256)
11020 POKE830,Y3:POKE831,0
11030 POKE832,X4-INT(X4/256)*256
```

```
11040 POKE833,INT(X4/256)
11050 POKE834,Y4:POKE835,0
11060 POKE786,AD
11070 POKE879,PC
11080 SYS(29158)
11090 RETURN
12000 POKE89,X-INT(X/256)*256
12010 POKE90,INT(X/256)
12020 POKE91,Y
12030 POKE784,AD
12040 POKE879,PC
12050 SYS(28782)
12060 RETURN
13000 FOR I=1 TO LEN(A$)
13010 B$=MID$(A$,I,1)
13020 POKE866,X-INT(X/256)*256
13030 POKE867,INT(X/256)
13040 POKE868,Y:POKE869,0
13050 POKE870,ASC(B$)
13060 POKE871,RN
13070 POKE879,PC
13080 POKE784,AD
13090 SYS(29749)
13100 X=X+XX:Y=Y+YY
13110 NEXT I
13120 RETURN
READY.
```

SPRITES

SPRITE THEORY

Sprites are hi-resolution blocks 24 pixels wide by 21 pixels high. The Sprite is stored in a 63 byte area of memory. As only one bank of 16K of memory may be addressed at any one time, using the video chip, a total of 240 Sprites may be defined (1K of that 16K memory is reserved for the screen). Although 240 Sprites may be defined, only 8 of those Sprites can be displayed on the screen at one time.

The Sprite may be moved anywhere within the screen and will not disturb any display already on the screen, i.e. it passes over (or under) the display. Sprites may be moved in distances of one pixel, giving the effect of continuous movement.

Sprites are controlled using the 47 registers of the video chip starting at address 53248($D000). Below is a diagram of these registers.

ADDRESS	DESCRIPTION
00 ($00)	Sprite 0 X position
01 ($01)	Sprite 0 Y position
02 ($02)	Same as 0 and 1 for Sprites 1-7
⋮	
15 ($0F)	
16 ($10)	M.S.B. of X position. SP0 in bit 0, SP1 in bit 1 etc.
18)$12)	Raster Register
19 ($13)	Light Pen X co-ordinate
20 ($14)	Light Pen Y co-ordinate
21 ($15)	Sprite enable register. SP0 in bit 0, SP1 in bit 1 etc.
23 ($17)	Sprite expand in Y. SP0 in bit 0, SP1 in bit 1 etc.
24 ($18)	Memory pointers, bit 0 not used
25 ($19)	Interrupt Register, bits 4, 5, and 6 not used
26 ($1A)	Enable Interrupt, bits 4, 5, 6, and 7 not used
27 ($1B)	Sprite-data priority. SP0 in bit 0, SP1 in bit 1 etc.
28 ($1C)	Sprite Multi-colour Select. SP0 in bit 0, SP1 in bit 1 etc.
29 ($1D)	Sprite expand in X. SP0 in bit 0, SP1 in bit 1 etc.
30 ($1E)	Sprite-Sprite collision. SP0 in bit 0, SP1 in bit 1 etc.
31 ($1F)	Sprite-Data collision. SP0 in bit 0, SP1 in bit 1 etc.
32 ($20)	Border colour
33 ($21)	Background colour #0
34 ($22)	Background colour #1
35 ($23)	Background colour #2
36 ($24)	Background colour #3
37 ($25)	Sprite Multi-colour #0
38 ($26)	Sprite Multi-colour #1
39 ($27)	Sprite 0 colour
40 ($28)	Same for Sprites 1-7

.
.
.
.
46 ($2E)

In registers 32-46, bits 4, 5, 6 and 7 are not used.

Two 'POKE's' are required to enable one Sprite. The first 'POKE' tells the computer which data block to look at when displaying the Sprite. This is done by 'POKE 2040+sp#,bl#', where sp# is the Sprite number (0 – 7) and bl# is the Sprite data block number (0 – 255). The starting location of the block is 64*bl#.

The second 'POKE' is to enable the Sprite, this is done by 'POKE 53269, PEEK (53269) OR2 (to the power of) sp #.

The Sprite is not yet on the screen, as it must be given a colour (or colours in Multi-colour Mode) and X and Y co-ordinates.

First, the colour; this is done by 'POKE 53287+sp#,col' where col is the colour from 0 – 15.

Then give the Sprite co-ordinates; this is done by:
'POKE 53248+sp#*2,Xand255'
'POKE 53248+sp#*2+1,Y'

If the X co-ordinate is larger than 255, then a third 'POKE' is required. This is:
'POKE 53264,PEEK(53264)OR2(to the power of)sp#

These 'POKE's' are useless unless a Sprite has previously been defined. This can be done in two different ways. One method is to take a sheet of graph paper and plot the points onto the paper. When done, each of the 21 lines has to be split up into three equal columns of eight points and a value calculated from these.

When this has been done for all of the now 63 bytes, the data can be entered into the machine in consecutive locations, starting from the first block location.

The second less tedious method is to use a simple Sprite editor. This has been developed to allow the user to use the screen as graph paper, the same program writes the Sprite as data at the end of the editor. More information about the Sprite editor can be found in the description of the program.

At the beginning of this section we mentioned that a maximum of 240 Sprites may be defined at any one time in the one memory bank. This is impossible in memory bank #0, since it is used for pointers and important variables, in addition, it is the bottom end of the basic programming area. This can be overcome by selecting another bank. An easy process allowing the user to have lots of Sprites available. The ideal bank to select is bank 2, this is the only bank, other than bank 0, that has a character set available. The process for selecting this memory bank is as follows:
'POKE 56578,PEEK(56578)OR 3'
'POKE 56576,(PEEK(56576)AND 252)OR 1'
'POKE 648,60'
'POKE 53272,(PEEK(53272)AND 15)OR 240'

In order, this routine sets bits 0 and 1 in the CIA #2 register to output, selects memory bank 2, and puts the video screen at the top end of that bank. The screen now resides at location 48128 – 49129, the Sprite pointers are now from 49144 to 49151. Sprites can be stored from block number 0 to 239 (the new location calculation is 64*bl#+32768).

The section of memory being used is still within the Basic programming area, this must now be protected by lowering the top of basic pointers and the string storage pointers. This is done by: 'POKE 56,128:POKE 53,128:CLR:NEW'

The co-ordinates of the two Sprites should be slightly different if they are next to each other. If the Sprites are unexpanded, the difference in the x co-ordinates is 24 and in the y direction is 21. For expanded mode the difference should be doubled. By having more than one Sprite defined in memory and there being a slight difference between the Sprites, dynamic displays may be produced by switching between one Sprite and another in a timed loop. This can be done by changing the value in one of the Sprite block pointers (2040-7) between the block numbers of the different Sprites. By using varying time loops, rapidly changing and/or slower displays may be created.
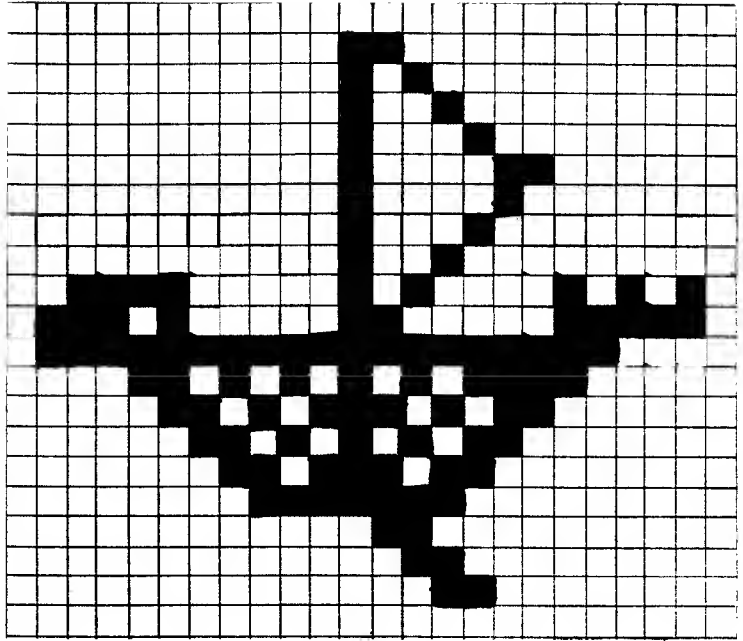
SPRITE GENERATOR

DESCRIPTION

The Sprite Generator facilitates the editing of up to 32 Sprites in the 64's memory. The Sprite being edited is displayed on the screen. If a point is on, there is a large dot, if off, a small dot. The actual Sprite is displayed in the bottom right hand corner of the screen. Options for editing the Sprite are displayed on the screen continuously. The program allows only the editing of standard hi-res Sprites, not Multi-colour (it should not be too difficult to convert it to Multi-colour use). Sprites are stored in the top end of memory bank 0, the video screen is in the normal place. The display is set up to emulate a large piece of graph paper and a coloured cursor shows the position on the grid, where any changes can be made. The Sprite is changed by positioning the cursor at the correct point and pressing '+' to turn that point on, or '-' to turn that point off.

RUNNING THE PROGRAM

The program will display, when run, the title as a Sprite. This may be cleared either by pressing 'SHIFT/CLR-HOME' to start editing or just hitting 'N' to move to the next Sprite. If a lower number Sprite is required, press 'E' in response to the question of which Sprite number. Entering the number (0-31) will then make that number the present Sprite. To store the Sprite press 'B', and the computer will write the Sprite values into DATA statements at the end of the program and return to the current Sprite. Pressing 'C' will allow a choice of colour for the Sprite (0-15). 'M' allows the user to move the Sprite around the screen under control of the keyboard. 'X' followed by either 'x' or 'y' (for x or y co'ordinate expand) will change the status – if expanded, it will make the Sprite smaller and vice-versa. Finally pressing 'Q' will quit the program.

   After editing and writing the Sprite as data, listing 30000 – will show the data for the edited Sprites. This has to be saved or the Sprite will be lost.

PROGRAM STRUCTURE

A brief run down of the lines is as follows:

Lines 90-100 : Check to see if there is any data to read in.
140-220 : Set up Sprite pointers, positions and variables.
260-350 : Display Sprite on the screen.
440-680 : Command link for keyboard input.
720-790 : Add a point.
830-910 : Go to a specific Sprite.

950-1030	:	Delete a point.
1070-1110	:	If any Sprites, enter them into memory.
1150-1180	:	Set up an array of powers of 2.
1220-1330	:	Input for expand in x or y.
1370-1520	:	Display control options.
1560-1610	:	Clear present Sprite.
1650-1850	:	Loop to move Sprite around the screen.
1890-1950	:	Change colour of present Sprite.
2000-2150	:	Add data to the end of the program.
2190-2290	:	Data for the title of program.

```
10 REM SPRITE GENERATOR
12 REM
14 REM
20 POKE 829,223
29 REM
30 REM ·IF ANY SPRITE DATA,SET UP SPRITE
31 REM
40 POKE 828,0
50 READ SP
60 IF SP>0THEN 810
69 REM
70 REM NO MORE SPRITE DATA
71 REM
80 GOSUB 860:POKE53281,2:POKE53280,2
90 DEFFNA(ZZ)=1064+R*40+C
100 V=53248:NO=PEEK(829)
110 XL=0:YL=1:XG=16:SE=21:XY=23:XX=29
120 SC=39:PRINT"⊐"
130 POKE 2040,NO:POKE V+SE,1:POKE V+XY,1
140 POKE V+XX,1:POKE V+XL,255:POKE V+YL,190
150 POKE V+XG,0
160 X=255:Y=190
169 REM
170 REM SET UP DISPLAY
171 REM
180 PRINT"⋇▓          ▌        ▌        ▐"
185 LOC=64*NO:PRINT"▓"
190 FORI=LOCTOLOC+62STEP3
200 FORJ=0TO2
210 ZZ=PEEK(I+J)
220 FORK=7TO0STEP-1
230 A=INT、(ZZANDA%(K))/A%(K))
240 IFA=1THENPRINT"◼●"::GOTO260
250 PRINT"◼.";
260 NEXTK
270 NEXTJ
280 PRINT
290 NEXTI
300 GOSUB1000
309 REM
310 REM SPRITE SET UP ON THE SCREEN
320 REM INPUT CHANGES
321 REM
330 R=0:C=0
340 Z=FNA(0)
350 POKEZ+54272,0
360 GETA$:IFA$=""THEN360
370 POKEZ+54272,1
380 IFA$="Q"THENPRINT"⊐":POKEV+21,0:PRINT"⊐"::E
ND
390 IFA$="▎"ANDC=23THEN C=0:GOTO340
```

78

```
400 IFA$="■"THENC=C+1:GOTO340
410 IFA$="■"ANDC=0THENC=23:GOTO340
420 IFA$="■"THENC=C-1:GOTO340
430 IFA$="▥"ANDR=20THENR=0:GOTO340
440 IFA$="▥"THENR=R+1:GOTO340
450 IFA$="⊓"ANDR=0THENR=20:GOTO340
460 IFA$="⊓"THENR=R-1:GOTO340
470 IFA$="▨"THENR=0:C=0:GOTO340
480 IFA$="⊓"THENGOSUB1150:GOTO340
490 IFA$="+"THEN580
500 IFA$="-"THEN730
510 IFA$="M"THEN1210
520 IFA$="B"THEN1450
530 IFA$="C" THEN 1400
540 IFA$="X"THEN900
550 IFA$="N"ANDNO-223<31THENNO=NO+1:GOTO130
560 IFA$="E"THEN660
570 GOTO 340
574 REM
575 REM ADD POINT
576 REM
580 Z=FNA(0)
590 Z1=PEEK(Z)
600 IFZ1=81THEN340
610 POKEZ,81
620 BYTE=INT(C/8)+R*3
630 BIT=7-(C-INT(C/8)*8)
640 POKEBYTE+NO*64,PEEK(BYTE+NO*64)ORA%(BIT)
650 GOTO 340
654 REM
655 REM INPUT SPRITE # TO EDIT
656 REM
660 INPUT"▩◖◖◖◖◖◖◖◖◖◖◖◖◖◖◖◖◖◖◖◖◖◖◖◖◖◖◖◖●●●●●●●■:SPRIT
E NO.        ■■■■■";S
670 IFS<0ORS>31THEN660
680 IF NO=223+STHENZZ=1:GOTO700
690 NO=223+S
700 PRINT"▩◖◖◖◖◖◖◖◖◖◖◖◖◖◖◖◖◖◖◖◖◖◖◖◖◖◖◖◖◖◖
                   ▨";
710 IFZZ=1THENZZ=0:GOTO340
720 GOTO 130
724 REM
725 REM DELETE POINT
726 REM
730 Z=FNA(0)
740 Z1=PEEK(Z)
750 IFZ1=46THEN 340
760 POKE Z,46
770 BYTE=INT(C/8)+R*3
780 BIT=7-(C-INT(C/8)*8)
790 POKE BYTE+NO*64,PEEK(BYTE+NO*64)AND(255-A%(
```

```
BIT>>
  800 GOTO 340
  804 REM
  805 REM IF ANY DATA, SET SPRITES UP
  806 REM
  810 LOC=SP*64
  820 FOR I=LOC TO LOC+62
  830 READ A:POKE I,A
  840 NEXT I
  850 GOTO 50
  854 REM
  855 REM SET ARRAY WITH POWERS OF TWO
  856 REM
  860 FOR I=0 TO 7
  870 A%(I)=2↑I
  880 NEXT I
  890 RETURN
  894 REM
  895 REM INPUT FOR EXPAND
  896 REM
  900 PRINT"🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵■■■■■■■■■■■EN
TER X OR Y"
  910 GETA$:IFA$<>"X"ANDA$<>"Y"THEN900
  920 IFA$="X"THEN960
  930 IFPEEK(V+XY)=1THENPOKEV+XY,0:GOTO980
  940 POKEV+XY,1
  950 GOTO980
  960 IFPEEK(V+XX)=1THENPOKEV+XX,0:GOTO980
  970 POKEV+XX,1
  980 PRINT"🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵🖵■■■■■■■■■■
                "
  990 GOTO 340
  994 REM
  995 REM DISPAY CONTROL OPTIONS
  996 REM
 1000 PRINT"🖵"SPC(26)"🖵CONTROLS■"
 1005 PRINTSPC(25)"SPRITE #🖵■"NO-223
 1010 PRINT:PRINTSPC(25)"🖵E■DIT SPRITE #"
 1020 PRINTSPC(25)"🖵N■EXT SPRITE #"
 1030 PRINTSPC(25)"🖵M■OVE SPRITE"
 1040 PRINTSPC(25)"🖵C■OLOUR CHANGE"
 1050 PRINTSPC(25)"🖵X■PAND"
 1060 PRINTSPC(25)"🖵+■ ADD DOT"
 1070 PRINTSPC(25)"🖵-■ REMOVE DOT"
 1080 PRINTSPC(25)"🖵B■ASIC DATA"
 1090 PRINTSPC(25)"🖵Q■UIT"
 1100 PRINT:PRINTSPC(25)"USE CURSOR"
 1110 PRINTSPC(25)"CONTROL TO"
 1120 PRINTSPC(25)"POSITION"
 1130 PRINTSPC(25)"CURSOR."
 1140 RETURN
```

```
1144 REM
1145 REM CLEAR PRESENT SPRITE
1146 REM
1150 FORI=0TO62:POKENO*64+I,0:NEXTI
1160 FORI=0TO20
1170 FORJ=0TO23
1180 POKE1064+I*40+J,46
1190 NEXTJ,I:R=0:C=0
1200 RETURN
1204 REM
1205 REM MOVE SPRITE AROUND SCREEN
1206 REM
1210 PRINT"◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙USE CURSOR K
EYS TO MOVE THE SPRITE."
1220 PRINT"◙RETURN TO RETURN TO EDITING"
1230 GETA$:IFA$=""THEN1230
1240 IFA$="◙"ANDX<319THENX=X+2
1250 IFA$="◙"ANDX>1THENX=X-2
1260 IFA$="◙"ANDY<254THENY=Y+2
1270 IFA$="◙"ANDY>1THENY=Y-2
1280 POKE V+YL,Y
1290 POKE V+XG,INT(X/255)
1300 POKE V+XL,X-INT(X/255)*255
1310 IF A$=CHR$(13)THEN1330
1320 GOTO1210
1330 POKE V+XL,255
1340 POKE V+YL,190
1350 POKE V+XG,0
1360 X=255:Y=190
1370 PRINT"◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙
                    "
1380 PRINT"
◙"
1390 GOTO 340
1394 REM
1395 REM CHANGE SPRITE COLOUR
1396 REM
1400 INPUT"◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙███████CO
LOUR (0-15)          ██████":CO
1410 IF CO<0ORCO>15THEN1400
1420 POKE V+SC,CO
1430 PRINT"◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙◙
                         ◙";
1440 GOTO 340
1444 REM
1445 REM CREATE DATA STATEMENTS FOR
1446 REM PRESENT SPRITE
1447 REM
1450 PRINT"◙◙◙◙";PEEK(828)+30000;"DATA"RIGHT$(S
TR$(NO),LEN(STR$(NO))-1)
1460 POKE828,PEEK(828)+1:FORI=0TO8
```

```
1470 PRINTPEEK(828)+30000"DATA";
1480 FORJ=0TO6
1490 BB=PEEK(NO*64+I*7+J)
1500 BB$=RIGHT$(STR$(BB),LEN(STR$(BB))-1)
1510 PRINTBB$;",";
1520 NEXT J
1530 PRINT"▓ ";POKE828,PEEK(828)+1
1540 NEXT I
1550 PRINTPEEK(828)+30000;"DATA-1"
1560 PRINT"RUN80▓"
1570 POKE 198,12
1580 FORI=0TO11;POKE631+I,13;NEXT I
1590 POKE829,NO;END
20000 DATA223
20001 DATA238,231,119,138,146,36,238
20002 DATA226,38,40,162,36,232,151
20003 DATA39,0,0,0,236,238,238
20004 DATA138,68,169,202,68,174,138
20005 DATA68,170,236,228,233,0,0
20006 DATA0,0,0,0,0,234,0
20007 DATA0,138,0,14,238,0,0
20008 DATA162,0,0,226,0,0,0
20009 DATA0,0,0,0,255,255,255
29997 REM
29998 REM SPRITE DATA STORED FROM HERE
29999 REM
30001 DATA -1
READY.
```

SPRITE DEMO

DESCRIPTION

Sprite Demo uses two Sprites to create a display. The Sprites were created using the Sprite Generator program. The Data statements were 'screen merged' into the demo program after removing them from the editor by listing the Data lines and deleting the editor. As the name suggests, this is only a demonstration program. It makes full use of the colour set, expand, and movement functions.

RUNNING THE PROGRAM

To run the program, load, and type 'RUN'. The display will appear after a couple of seconds since the Sprite Data must be read into memory from the Data lines at the end of the program. When the display appears follow the instructions to expand the display, change the colour, and move it about the screen under control of the cursor keys.

PROGRAM STRUCTURE

A brief synopsis of the program lines is as follows:

Lines 10-80	: Read the Sprite data into memory at the required block.
110-140	: Set up Sprite pointers.
150-160	: Put Sprite on the screen.
170	: Enables Sprites 0 and 1.
180-200	: Wait for key press.
210-270	: Disable Sprites, enlarge, reposition and re-enable.
280-340	: Loop to change colour of Sprites.
350-460	: Loop to move Sprites under control of the cursor keys.
470-490	: Turn off Sprites and end program.
30000-30020	: Data for Sprites.

```
10 FORI=0TO1
20 READ NO
30 FORJ=0TO62
40 READ X
50 POKENO*64+J,X
60 NEXT J
70 POKE2040+I,NO
80 NEXT I
90 PRINT"⊐▨ DEMONSTRATION OF HOW TO USE MORE THAN ";
100 PRINT"▨    ONE SPRITE TO CREATE A DISPLAY    ▨"
110 V=53248:REM START OF VIDEO CHIP
120 POKEV+32,12:POKEV+33,12:REM SCREEN AND BORDER COLOURS
130 POKEV+23,0:POKEV+29,0:REM X AND Y SMALL
135 POKEV+39,2:POKEV+40,2:REM SPRITE COLOURS ARE RED
140 X=150:Y=150:REM COORDINATES OF SPRITES
150 POKEV,X:POKEV+1,Y
160 POKEV+2,X+24:POKEV+3,Y
170 POKEV+21,3
180 PRINT"▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨"
190 PRINT" ▨HIT ANY KEY TO SEE ENLARGED PICTURE   ▨"
200 GETA$:IFA$=""THEN200
210 POKEV+21,0
220 PRINT"▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨            ▨"
230 PRINT"
235 X=126:Y=140
240 POKEV+23,3:POKEV+29,3:REM X AND Y ENLARGED
250 POKEV:X:POKEV+1,Y
260 POKEV+2,X+48:POKEV+3,Y
270 POKEV+21,3
280 PRINT"▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨"
290 PRINT" ▨▨ CHOOSE A COLOUR (0-9, RET TO EXIT   ▨"
300 GETA$:IFA$=""THEN300
310 IFA$=CHR$(13)THEN350
320 A=VAL(A$)
330 POKEV+39,A:POKEV+40,A
340 GOTO300
350 PRINT"▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨"
360 PRINT" ▨MOVE THE PICTURE WITH THE CURSOR KEYS"
370 PRINT" ▨    HIT RETURN TO END DEMO           ▨"
380 GETA$:IFA$=""THEN380
```

84

```
390 IFA$="X"ANDY<255THENY=Y+1:GOTO440
400 IFA$="]"ANDY>0THENY=Y-1:GOTO440
410 IFA$="N"ANDX<207THENX=X+1:GOTO440
420 IFA$="I"ANDX>0THENX=X-1:GOTO440
425 IFA$=CHR$(13)THEN470
430 GOTO380
440 POKEV,X:POKEV+1,Y
450 POKEV+2,X+48:POKEV+3,Y
460 GOTO380
470 POKEV+21,0:PRINT"XX";
480 POKEV+33,6:POKEV+32,14
490 END
30000 DATA224
30001 DATA0,0,0,0,0,0,0
30002 DATA0,0,0,0,0,0,0
30003 DATA0,0,0,0,0,0,3
30004 DATA0,0,7,31,255,131,15
30005 DATA255,192,7,255,128,0,28
30006 DATA0,0,28,0,0,14,0
30007 DATA0,14,0,0,7,1,0
30008 DATA7,1,0,31,255,0,15
30009 DATA255,0,7,255,0,0,0
30010 DATA225
30011 DATA0,0,0,0,0,0,0
30012 DATA0,0,0,192,0,3,240
30013 DATA0,7,248,0,255,255,240
30014 DATA255,255,248,255,255,240,56
30015 DATA192,0,56,0,0,112,0
30016 DATA0,112,0,0,224,0,0
30017 DATA224,0,0,192,0,0,192
30018 DATA0,0,252,0,0,254,0
30019 DATA0,252,0,0,0,0,0
30020 DATA-1
READY.
```

CHARACTER THEORY

### USER DEFINED CHARACTERS ON THE 64

Sprites on the 64 provide the means to define shapes and move them about the screen with reasonable ease, but they have their limitations. Only 8 Sprites can be displayed on the screen at any one time. If a shape the size of an ordinary character is needed, a lot of superfluous data is involved in defining the Sprite. In addition, Sprites are not obtainable directly from the keyboard, they have to be presented, coloured, and moved with a series of POKE statements or the machine code equivalent.

For those users who need to define their own shapes which are normal characters and directly usable from the keyboard, then user definable characters offer a solution. It is possible to replace, by software, the entire set of characters resident in the CBM 64.

However, for most purposes, a number of the normal characters (such as the letters and figures) will probably be required in addition to the shapes the user may design himself. The reversed image of these characters is the area best suited for the user to place his own designed shapes, thereby leaving the normal character set available.

Every character on the 64 is designed on a grid of 8 by 8 bits giving a block of 64 bits per character, each bit is called a pixel. If the user examines the two characters @ and A on their grids they would appear as in fig. a.

fig. a
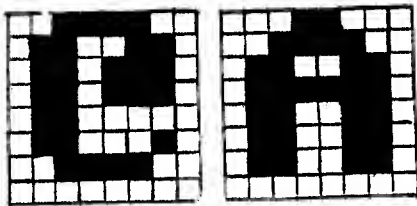
Note that with these characters, the bottom, left, and right hand edges of the grid have been left empty. This is to prevent adjacent characters merging into each other when they are printed on the screen. Graphics characters can go right to the edge of the grid, they will then join up with each other, enabling them to be used to create pictures.

To define characters, first draw them on an 8 x 8 grid as shown in fig. b.

86

fig. b
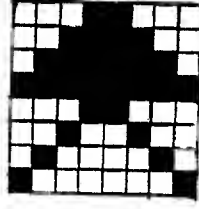
All the squares which are filled in, will be switched 'on', and are therefore bright points on the screen, whereas the empty squares will be 'off' and thus dark and invisible. The user will have to tell the computer which squares to switch on and which to switch off. This is accomplished by giving each column of the grid a value between 1 and 128 as shown in fig. c.
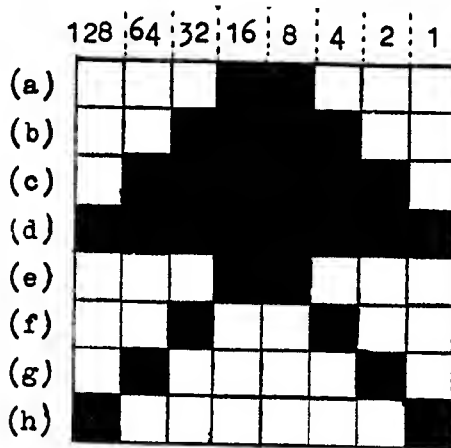
fig. c

Look at the top row (a) of the grid and add together the values of all the squares which are to be 'on'. In this particular case the result is 24. Repeat this for all the remaining rows (b to h).

There are now 8 values, and these are shown in the table below.

Row (a) 24
Row (b) 60
Row (c) 126
Row (d) 255
Row (e) 24
Row (f) 36
Row (g) 66
Row (h) 129

These 8 values are needed to inform the computer which bits (pixels) should be switched 'on' and which should be 'off'.

Having designed the character on paper and calculated the necessary values from the drawing the user may now turn to the 64.

The shape of the resident characters in the 64 is stored in ROM, and as such cannot be altered by software. However, there is nothing to prevent the user copying this ROM into RAM where it can easily be modified.

The ROM containing the shapes of the 64's resident characters is to be found at address 53248 ($D000) to 57343 ($DFFF). Because of the way the 64 is configured, it is not possible to access this ROM directly. It is sandwiched between the I/O ROM (sitting at identical memory addresses on the top of it) and RAM (underneath it). Uncovering the character ROM by switching away the overlying I/O ROM can easily be accomplished with POKE 1,51. However, the system interrupts expect to find the I/O in place, so the user must first disable the interrupts with POKE 56333,127. If this is not done, the machine will crash. The first two lines of a program to enable user defined characters are now ready.

Having released the character ROM for access, the character data can be copied into ROM. The characters transferred in the program are the first 256 characters from the character ROM. To do this:

FOR I-0 to 2047: REM 2048 BYTES
POKE 12288+I,PEEK(53248+I)
NEXT I

All the required characters are now in RAM starting at location 12288. The 64 is still, however, using the character ROM.

The next step is to restore the I/O and the interrupts by:

POKE 1,PEEK(1)OR4
POKE 56334,PEEK(56334)OR 1

The computer must then be instructed to use the new character set, and for this another POKE is required. This is POKE 53272,(PEEK(53272)AND240)+12. The new character set is now being used by the processor to generate the display. The only thing left to do is change the characters used, by changing the values in memory to those created in the grid design. If at any time the original character set is required, POKE 53272,21 will restore the pointer to the ROM image.

CHARACTER EDITOR

DESCRIPTION

This program facilitates the editing of up to 64 characters in the 64's memory. There are two sections to this program, the first section displays the characters that can be edited, the second section is the editing routine.

To choose a character to edit, move the cursor to the required character and enter either 'E' for edit or 'N' which clears the character and edits. When in edit mode, move the cursor using the cursor keys over the enlarged character and use the keys '+' and '-' to change a point. To write the character as data, enter 'B', and when the character has been updated, typing 'R' will return to the first display where the newly edited character is displayed in full size.

RUNNING THE PROGRAM

The editing of characters is like the Sprite editor except for one difference, the character is not changed in memory unless the '=' key is pressed, updating the character. When run, the program firstly copies the character ROM into RAM. There is a long pause whilst this is happening, the display will then appear, after which just follow the controls displayed on the screen.

PROGRAM STRUCTURE

A brief run down of the program lines is as follows:

130-320	:	Copy the characters into RAM.
330-350	:	Define functions for editing.
360-400	:	Display 'N' character.
410-610	:	Input from keyboard and cursor control.
710-770	:	Commands for editing character.
810-840	:	Commands for choosing character.
1010-1170	:	Display character set options.
1210-1300	:	Display editing options.
1520-1540	:	End program.
1610-1700	:	Update character.
1810-1910	:	Edit character.
2010-2160	:	Add 'Basic' data statements to the end of program.

```
1 REM CHARACTER BUILDING
2 REM **********************
3 REM
130 POKE 53280,2:POKE 53281,2
140 PRINT"        * CHARACTER BUILDING *"
150 POKE 828,0
160 RUN 170
170 CS=12288
175 POKE 56334,PEEK(56334)AND254:POKE 1,PEEK(1)AND251
180 FOR I=CS TO CS+2047
190 POKE I,PEEK(53248+I-CS)
200 NEXT I
205 POKE 1,PEEK(1)OR4:POKE 56334,PEEK(56334)OR1
210 PRINT" RUN 280"
220 PRINT"RUN"
230 POKE 198,3
240 POKE 631,19
250 POKE 632,13
260 POKE 633,13
270 END
280 S=1024:CL=40
290 CS=12288
300 CR=0:LN=30000+PEEK(828)
310 P=24:BG=1:BR=1
320 POKE 53280,2:POKE 53281,2
330 DEFFNA(XX)=S+R*2*CL+2*C:REM SCREEN POKE LOCATION
340 DEFFNB(XX)=8*R+C:REM SCREEN POKE VALUE FOR CHAR
350 GOTO 1000
360 PRINT"  ":GOSUB 1200
370 PRINT"  ";:FOR I=0 TO 7
380 PRINT". . . . . . . .":PRINT
390 NEXT I:F=0
400 PRINT"  ":R=0:C=0
410 Z=FNA(0)
420 IF F=0 THEN 460
430 IF Z=ZL THEN 450
440 POKE ZL,IL:ZL=Z:IL=PEEK(ZL)
450 POKE Z+54272,0
460 POKE Z+54272,0
470 GET A$:IF A$="" THEN 470
480 POKE Z+54272,1
490 REM
500 REM CURSOR CONTROL OPTIONS
505 REM
510 IF A$="Q" THEN 1500
520 IF A$="M" AND C=7 THEN C=0:GOTO 410
```

```
530 IF A$="▊" THEN C=C+1:GOTO 410
540 IF A$="▊" AND C=0 THEN C=7:GOTO 410
550 IF A$="▊" THEN C=C-1:GOTO 410
560 IF A$="▊" AND R=7 THEN R=0:GOTO 410
570 IF A$="▊" THEN R=R+1:GOTO 410
580 IF A$="▊" AND R=0 THEN R=7:GOTO 410
590 IF A$="▊" THEN R=R-1:GOTO 410
600 IF A$="▊" THEN 400
610 IF F=1 THEN 800
695 REM
700 REM DEFINE NEW CHARACTER OPTIONS
705 REM
710 IF A$="+" THEN POKE Z,81:GOTO 410
720 IF A$="-" THEN POKE Z,46:GOTO 410
730 IF A$="=" THEN 1600
740 IF A$="▊" THEN 370
750 IF A$="R" THEN 1000
760 IF A$="B" THEN 2000
770 GOTO 410
795 REM
800 REM REVIEW CHARACTER SET OPTIONS
805 REM
810 CR=FNB(0)
820 IF A$="N" THEN POKE 53272,21:GOTO 360
830 IF A$="E" THEN POKE 53272,21:F=0:GOTO 1800
840 GOTO 410
995 REM
1000 REM DISPLAY CHARACTER SET OPTIONS
1005 REM
1010 POKE 53272,(PEEK(53272)AND240)+12:R=4:C=0
1020 ZL=FNA(0):IL=32
1030 F=1:PRINT"▊":
1040 PRINT"▊▊ A B C D E F G":PRINT
1050 PRINT"H I J K L M N O":PRINT
1060 PRINT"P Q R S T U V W":PRINT
1070 PRINT"X Y Z [ £ ] ↑ ←":PRINT
1080 PRINT" ! "CHR$(34)" # $ % & '":PRINT
1090 PRINT"( ) * + , - . /":PRINT
1100 PRINT"0 1 2 3 4 5 6 7":PRINT
1110 PRINT"8 9 : ; < = > ?":PRINT
1120 PRINT"▊"SPC(25)"▊OPTIONS▊":PRINT
1130 PRINTSPC(22)"▊N NEW CHAR▊":PRINT
1140 PRINTSPC(22)"▊E EDIT CHAR▊":PRINT
1150 PRINTSPC(22)"▊Q QUIT▊"
1160 BC=PEEK(55296)
1170 GOTO 410
1195 REM
1200 REM EDIT OPTIONS
1205 REM
1210 PRINT"▊▊"SPC(25)"▊OPTIONS▊":PRINT
1220 PRINT
```

92

```
1230 PRINTSPC(P)"#+# ADD DOT":PRINT
1240 PRINTSPC(P)"#-# ERASE":PRINT
1250 PRINTSPC(P)"#=# UPDATE":PRINT
1260 PRINTSPC(P)"#R# REVIEW":PRINT
1270 PRINTSPC(P)"#Q# QUIT":PRINT
1280 PRINTSPC(P)"#B# ADD DATA":PRINT
1290 PRINTSPC(P+1)"]STATEMENT"
1300 RETURN
1495 REM
1500 REM QUIT
1505 REM
1510 REM
1520 POKE 53272.21
1530 POKE 53281.6:POKE 53280.14
1540 PRINT"XX    BYE!"
1550 END
1595 REM
1600 REM UPDATE
1605 REM
1610 PRINT"X":
1620 X=CS+8*CR
1630 FOR R=0 TO 7:SM=0
1640 FOR C=0 TO 7:D=7-C
1650 SM=SM-2↑D*(PEEK(FNA(0))=81)
1660 NEXT C
1670 POKE X+R.SM
1680 PRINTSPC(17):SM:PRINT
1690 NEXT R:R=0:C=0
1700 GOTO 410
1795 REM
1800 REM EDIT CHAR
1805 REM
1810 PRINT"]"
1820 X=CS+8*CR
1830 FOR R=0 TO 7:Y=PEEK(X+R)
1840 FOR C=0 TO 7:Z=FNA(0)
1850 Q=46:Y=Y*2
1860 IF Y>255 THEN Q=81:Y=Y-256
1870 POKE Z.Q:POKE Z+54272,1
1880 NEXT C.R
1890 R=0:C=0
1900 GOSUB 1200
1910 GOTO 410
1995 REM
2000 REM ADD DATA STATEMENTS
2005 REM
2010 X=CS+8*CR
2020 PRINT"]▓▓▓▓▓▓▓▓"
2030 PRINTLN;"DATA":
2040 PRINTRIGHT$(STR$(X).LEN(STR$(X))-1);
2050 FOR I=X TO X+7
```

```
2060 PRINT",";
2070 PRINTRIGHT$(STR$(PEEK(I)),LEN(STR$(PEEK(I)))-1);
2080 NEXT I
2090 PRINT:PRINT"RUN ▧"
2100 POKE 828,PEEK(828)+1
2110 POKE 198,9
2120 FOR I=0 TO 8
2130 POKE I+631,13
2140 NEXT I
2160 END
READY.
```

94

DISPLAY MANAGEMENT

INTRODUCTION

The routines that follow in this chapter are for inclusion within programs written by the user. They are very useful for setting up screen displays.

This chapter is split into four sections. Each section concerns one aspect of producing screen displays with machine code routines to implement them. These routines are in the form of basic loaders that must be run to use the example programs following them.

Some of the loaders in the later sections will contain routines from previous sections. This is because they are required for the demonstration programs to work.

All the routines that follow can be entered into memory at the same time without any of them over-lapping.

DISPLAY MANAGEMENT

THE FIRST ROUTINE

The first Machine Code routine in this section is used by most of the other routines to calculate the 64's screen addresses; the addresses are returned in x and y. This routine is never called from a Basic program, only by the other routines and should not be used on its own, as this will cause a Machine Crash. This routine will also· trap any errors and set an error flag.

SIMPLE DISPLAY ROUTINES

The first five routines in this section are the simplest routines in Display Management and provide an insight into the methods used in the more sophisticated routines in this section. All five routines rely on the initial calculation of the screen memory address of the first character in the line or block to be displayed, and this is usually the top left corner character of that line or block. The address is calculated from the line and column numbers which designate the co-ordinates of the character, using the first routine starting at $7000 (hex). All the display functions are performed by the calculation of screen memory addresses using the starting address as a reference point. The 1000 bytes of the screen memory in the 64 can be subdivided into 25 smaller blocks (lines), each of 40 locations.

Drawing a line from the top to the bottom of the screen in column 5 is simply a matter of filling every fifth location in each of the 25 sub blocks with the desired ASCII code value. Drawing a horizontal line across the screen simply means filling all the locations in one line block of memory with the specified character. The starting position within a sub block is determined by the starting screen address and the length of the line is determined by a variable transferred from the Basic program. Filling a block of the screen with a specified character simply means repeatedly drawing horizontal lines until the block is filled. To reverse field a line or a block of the screen, the character in each screen memory location has a logical Exclusive OR operation performed on it.

All the Machine Code routines have been turned into Basic loaders and presented as such in this book. They have also been programmed so that they are chained together and should be entered in ascending order of line numbers.

SCREEN ADDRESSES

The subroutine used to calculate the screen addresses from values in locations 89 for column and 90 for line, is only called by other routines and will cause the 64 to crash if called on its own. This routine also contains the error flag in location 785.

HORIZONTAL BAR

SYS (28800)
A horizontal bar of any 64 characters is drawn by this routine on the screen, with a single character space resolution. The position of the left hand end of the bar is determined by two variables: column number and line number. The length of the bar can be a integer from 1 to 255 character spaces, although the maximum line length will be equal to the screen width. A variable allows the character used to draw the line to be defined in the Basic calling program. The routine requires the following variable locations:

89 column number of horizontal line start (left end).

90 screen line number of horizontal line start.

91 length of horizontal line.

784 ASCII code value of characters used in horizontal line.

786 the screen colour is held in this location.

This routine is called from a Basic program with SYS (28800).

VERTICAL BAR

SYS (28832)
This routine draws a vertical bar of any 64 character on the screen with a single character space resolution. The position of the left hand end of the bar is determined by two variables: column number and line number. The length of the bar can be an integer from 1 to 255 character spaces, although the maximum line length will be equal to the screen width. A variable allows the character used to draw the line to be defined in the Basic calling program. The routine requires the following variable locations:

89 column number if vertical line start (top end).

90 screen line number of vertical line start.

91 length of vertical line.

784 ASCII code value of characters used in vertical line.

786 this is the colour location.

This routine is called from a Basic program with SYS (28832).

BLOCK FILL

SYS (28896)
This routine fills a designated block of the screen with a previously defined character. The position of the top left corner of the block is determined by two variables: column number and line number. The width of the block can be an integer from 1 to 255 character spaces. Also, the height of the block may be an integer value from 1 to 255, although in practice the size of the block is restricted to the screen dimensions. A variable allows the character used to fill the block which is to be defined in the Basic calling program. This routine requires the following variable locations:

89 column number of block start.
90 screen line number of block start.
91 width of block.
92 height of block.
784 ASCII code value of characters used in block.
786 the screen colour is held in this location.

This routine is called from a Basic program with SYS (28896)

INVERT

SYS (28960)
Invert reverses the field of all characters within a block of screen area. The position of the top left corner of the block is determined by two variables: column number and line number. The width of the block can be an integer from 1 to 255 character spaces. Also the height of the block may be an integer value from 1 to 255, although in practice the size of the reverse field is restricted to the screen dimensions. A variable allows the character used to fill the block to be defined in the Basic calling program. This routine requires the following variable locations:

89 column number of reverse block start (top left).
90 screen line number of reverse block start.
91 width of reverse block area.
92 height of reverse block area.
786 is the colour location.

This routine is called from a Basic program with SYS (28960).

BORDER

SYS (29024)

As its name implies, this routine will draw a thin line border on a specified area of the screen. The position of the top left corner of the border is determined by two variables: the column number and the line number. The width of the border can be an integer from 1 to 255 character spaces. The border height can also be any integer from 1 to 255 although the maximum size of the border that can be displayed is restricted to the outside edge of the screen. This routine requires the following variable locations:

89 column number of border start (top left).

90 screen line number of border start.

91 width of border.

92 height of border.

786 location of colour.

This routine is called with SYS (29024).

CURSOR CONTROL

The Machine Code cursor control routine, a part of this package, is called by SYS (29216), but can be anywhere in the memory, providing it is protected from being overwritten by Basic. The number of times the routine is called depends upon the values placed in locations 87 and 88. This routine uses a loop to print characters at specified points. If the user calls this routine on its own, he must POKE the appropriate values into locations 87 and 88.

POKE 87, x co-ordinate value

POKE 88, y co-ordinate value

```
10 REM
20 REM *****************************
30 REM *ROUTINE TO CALCULATE SCREEN   *
40 REM *ADDRESS FROM VALUES FOR       *
50 REM *     COLUMN     IN    89      *
60 REM *     LINE       IN    90      *
70 REM *ADDRESS RETURNED IN 91 AND 92*
80 REM *THIS ROUTINE IS ONLY CALLED   *
90 REM *BY OTHER ROUTINES IN THE      *
100 REM*PACKAGE.DO NOT USE BY ITSELF.*
110 REM*ERROR FLAG IN LOCATION 785    *
120 REM*****************************
130 I=28672:T=0
140 READ A
150 IF A=-1 THEN 190
160 T=T+A:POKEI,A
170 I=I+1
180 GOTO 140
190 PRINT"JROUTINE 1:"
200 IF T=11973 THEN PRINT"XENTERED O.K":GOTO 1000
210 PRINT"XENTERED INCORRECTLY"
220 END
230 DATA169.0.141.17.3.165.89
240 DATA48.66.201.40.176.74.165
250 DATA90.48.82.201.25.176.90
260 DATA169.0.133.87.133.88.133
270 DATA94.133.95.165.90.240.15
280 DATA170.24.165.87.105.40.133
290 DATA87.144.2.230.88.202.208
300 DATA242.24.165.87.101.89.133
310 DATA87.165.88.105.4.133.88
320 DATA24.165.87.133.94.165.88
330 DATA105.212.133.95.96.169.1
340 DATA141.17.3.169.0.133.89
350 DATA76.7.112.169.2.141.17
360 DATA3.169.39.133.89.76.7
370 DATA112.169.3.141.17.3.169
380 DATA0.133.90.76.7.112.169
390 DATA4.141.17.3.169.24.133
400 DATA90.76.7.112.-1
1000 REM
1010 REM
1020 REM****************************
1030 REM*DRAW A HORIZONTAL BAR OF ANY *
1040 REM*64 CHARACTER.CHARACTER STORED*
1050 REM*IN 784. COLUMN START IN 89.  *
1060 REM*LINE START IN 90. LENGTH OF  *
1070 REM*LINE IN 91. COLOUR IN 786.   *
1080 REM*ROUTINE CALLED BY SYS(28800) *
1090 REM****************************
1100 I=28800:T=0
```

```
1110 READ A
1120 IF A=-1 THEN 1160
1130 POKE I,A:T=T+A
1140 I=I+1
1150 GOTO 1110
1160 PRINT"⤓ROUTINE 2:"
1170 IF T=3095 THEN PRINT"▨▨ENTERED O.K":GOTO 2000
1180 PRINT"▨▨ENTERED INCORRECTLY"
1190 END
1200 DATA72,152,72,138,72,32,0
1210 DATA112,164,91,173,16,3,145
1220 DATA87,173,18,3,145,94,136
1230 DATA208,243,104,170,104,168,104
1240 DATA96,-1
2000 REM
2010 REM
2020 REM********************************
2030 REM*DRAW A VERTICAL BAR OF ANY 64*
2040 REM*CHARACTER. CHARACTER STORED  *
2050 REM*IN 784. COLUMN START IN 89,   *
2060 REM*LINE START IN 90. LENGTH OF   *
2070 REM*LINE IN 91. COLOUR IN 786.    *
2080 REM*ROUTINE CALLED BY SYS(28832)  *
2090 REM********************************
2100 I=28832:T=0
2110 READ A
2120 IF A=-1 THEN 2160
2130 POKE I,A:T=T+A
2140 I=I+1
2150 GOTO 2110
2160 PRINT"⤓ROUTINE 3:"
2170 IF T=5296 THEN PRINT"▨▨ENTERED O.K.":GOTO 3000
2180 PRINT"▨▨ENTERED INCORRECTLY"
2190 END
2200 DATA72,152,72,138,72,32,0
2210 DATA112,160,0,166,91,173,16
2220 DATA3,145,87,173,18,3,145
2230 DATA94,24,165,87,40,133
2240 DATA87,133,94,165,88,105,0
2250 DATA133,88,105,212,133,95,202
2260 DATA208,224,104,170,104,168,104
2270 DATA96,-1
3000 REM
3010 REM
3020 REM********************************
3030 REM*FILL A BLOCK OF THE SCREEN    *
3040 REM*WITH A SPECIFIED CHARACTER.   *
3050 REM*CHARACTER STORED IN 784. TOP  *
3060 REM*LEFT COORDINATES IN 89(COL)   *
3070 REM*AND 90(LINE). HEIGHT OF BLOCK*
3080 REM*IN 91. WIDTH IN 92,AND COLOUR*
```

```
3090 REM*IN 786.                       *
3100 REM*ROUTINE CALLED BY SYS(28896) *
3110 REM*********************************
3120 I=28896:T=0
3130 READ A
3140 IF A=-1 THEN 3180
3150 POKE I,A:T=T+A
3160 I=I+1
3170 GOTO 3130
3180 PRINT"]ROUTINE 4:"
3190 IF T=5588 THEN PRINT"XXENTERED O.K.":GOTO 4000
3200 PRINT"XXENTERED INCORRECTLY"
3210 END
3220 DATA72,152,72,138,72,32,0
3230 DATA112,166,92,164,91,136,173
3240 DATA16,3,145,87,173,18,3
3250 DATA145,94,136,16,243,24,165
3260 DATA87,105,40,133,87,133,94
3270 DATA144,4,230,88,230,95,202
3280 DATA208,222,104,170,104,168,104
3290 DATA96,-1
4000 REM
4010 REM*********************************
4020 REM*INVERT A BLOCK OF THE SCREEN *
4030 REM*TOP LEFT COORDINATES IN 89   *
4040 REM*(COL) AND 90(LINE), HEIGHT IN*
4050 REM*91 AND WIDTH IN 92, COLOUR IN*
4060 REM*786.                         *
4070 REM*ROUTINE CALLED BY SYS(28960) *
4080 REM*********************************
4090 I=28960:T=0
4100 READ A
4110 IF A=-1 THEN 4150
4120 POKE I,A:T=T+A
4130 I=I+1
4140 GOTO 4100
4150 PRINT"]ROUTINE 5:"
4160 IF T=5859 THEN PRINT"XXENTERED O.K.":GOTO 5000
4170 PRINT"XXENTERED INCORRECTLY"
4180 END
4190 DATA72,152,72,138,72,32,0
4200 DATA112,166,92,164,91,136,177
4210 DATA87,73,128,145,87,173,18
4220 DATA3,145,94,136,16,242,24
4230 DATA165,87,105,40,133,87,133
4240 DATA94,144,4,230,88,230,95
4250 DATA202,208,221,104,170,104,168
4260 DATA104,96,-1
5000 REM
5010 REM*********************************
5020 REM*DRAW A BORDER OF ANY SIZE,ANY*
```

```
5030 REM*LOCATION. AND ANY COLOUR. TOP*
5040 REM*LEFT COORDINATES ARE:       *
5050 REM*    COLUMN      IN    89     *
5060 REM*    LINE        IN    90     *
5070 REM*    WIDTH       IN    91     *
5080 REM*    HEIGHT      IN    92     *
5090 REM* COLOUR IS STORED IN 786.    *
5100 REM*ROUTINE CALLED BY SYS(29024) *
5110 REM*****************************
5120 I=29024:T=0
5130 READ A
5140 IF A=-1 THEN 5180
5150 POKE I,A:T=T+A
5160 I=I+1
5170 GOTO 5130
5180 PRINT":ROUTINE 6:"
5190 IF T=20557 THEN PRINT"XXENTERED O.K.":GOTO6000
5200 PRINT"XXENTERED INCORRECTLY"
5210 END
5220 DATA72,152,72,138,72,32,0
5230 DATA112,165,87,133,89,165,88
5240 DATA133,90,169,100,141,16,3
5250 DATA32,212,113,24,165,87,105
5260 DATA41,133,87,133,89,165,88
5270 DATA105,0,133,88,133,90,169
5280 DATA101,141,16,3,32,239,113
5290 DATA24,165,89,101,91,133,87
5300 DATA198,87,165,90,105,0,133
5310 DATA88,169,103,141,16,3,32
5320 DATA239,113,165,89,133,87,165
5330 DATA90,133,88,166,92,198,87
5340 DATA202,202,24,165,87,105,40
5350 DATA133,87,165,88,105,0,133
5360 DATA88,202,208,240,169,99,141
5370 DATA16,3,32,212,113,104,170
5380 DATA104,168,104,96,164,91,165
5390 DATA87,133,94,165,88,24,105
5400 DATA212,133,95,173,16,3,145
5410 DATA87,173,18,3,145,94,136
5420 DATA208,243,96,160,0,166,92
5430 DATA202,202,173,16,3,145,87
5440 DATA24,165,87,133,94,165,88
5450 DATA105,212,133,95,173,18,3
5460 DATA145,94,24,165,87,105,40
5470 DATA133,87,165,88,105,0,133
5480 DATA88,202,208,219,96,-1
6000 REM
6010 REM*****************************
6020 REM*ROUTINE TO PLACE THE CURSOR  *
6030 REM*AT A LOCATION ON THE SCREEN  *
6040 REM*WHOSE COORDINATES ARE STORED *
```

DEMO

The loader on pp. 101-5 should be loaded and run once before using this demo.

DESCRIPTION

This program is only intended as a demo of some of the Machine Code routines included in the Display Management section. The Code routines used are as follows:
CURSOR CONTROL – SYS (29216)
HORIZONTAL LINE – SYS (28800)
VERTICAL LINE – SYS (28832)
BORDER – SYS (29024)
BLOCK DISPLAY – SYS (28896)
BLOCK REVERSE – SYS (28960)

PROGRAM STRUCTURE

The lines of interest in the program are as follows:

130-145	: Limit memory and clear pointers.
214	: Set screen and border colours.
215-245	: Uses the Border routine to set up and display all the necessary borders on the screen.
295-335	: Sets up the vertical and horizontal bars.
400-405	: This uses the block reverse routine to display the character input line.
480-630	: These lines use the cursor control routine to set up the text on the screen. Each item of text is held as three elements, they are: column, line and text.
730-740	: This routine highlights the function currently selected from the menu.
780-800	: This routine removes the highlighted function when another is selected.
940-960	: This is the error message (record full), it is displayed using the cursor control routine.
1040-1070	: Highlights the description and places the cursor in the next available line awaiting an input.
1120-1140	: If the input is an up arrow then the program exits the current section and moves to the next.
1180-1185	: Removes the highlight from the description.
1250-1300	: Reverse field to highlight the Quantity and place cursor on the next available line awaiting input.
1340-1345	: Remove highlight from Quantity.
1410-1460	: Reverse field to highlight Cost and place cursor on

106

```
6050 REM*IN  87=LINE AND 88=COLUMN.   *
6060 REM*ROUTINE  CALLED BY SYS(29216)*
6070 REM*****************************
6080 I=29216:T=0
6090 READ A
6100 IF A=-1 THEN 6140
6110 POKE I,A:T=T+A
6120 I=I+1
6130 GOTO 6090
6140 PRINT"ROUTINE 7:"
6150 IF T=5163 THEN PRINT"ENTERED O.K.":GOTO 6240
6160 PRINT"ENTERED INCORRECTLY"
6170 END
6180 DATA72,152,72,138,72,169,19
6190 DATA32,22,231,165,87,240,9
6200 DATA169,17,32,22,231,198,87
6210 DATA208,247,165,88,240,9,169
6220 DATA29,32,22,231,198,88,208
6230 DATA247,104,170,104,168,104,96,-1
6240 POKE51,0:POKE52,112
6250 POKE55,0:POKE56,112:CLR:NEW
READY.
```

		the next available line awaiting input.
1500-1505	:	Remove highlight from Cost.
1560-1570	:	Reverse field to highlight Total.
1610-1620	:	Place cursor at position to display total.
1660-1665	:	Remove highlight from total.
1720-1750	:	Moves the cursor to the location for prompt and displays prompt.
1790-1810	:	Uses the Horizontal Bar routine to delete the prompt.
1880-1900	:	Calculates the highlights the grand total. The cursor routine is called to display the total.
1940-1945	:	Remove highlight from total.
2000-2010	:	Reverse to highlight name and address.
2060-2170	:	Move cursor to the first position on the name and address section.
2210-2215	:	Remove highlight from name and address.
2260-2270	:	Remove highlight from update and return to select.
2370-2380	:	Reverse to highlight delete function.
2240-2475	:	Using the Reverse block routine, delete blocks of text from the screen by replacing them with spaces.
2520-2525	:	Using horizontal line delete lines, by replacing them with a space character.

When you are entering the address, finish each line with a comma and then press the return key. To exit the address mode enter a full stop and press the return key.

```
10 REM ******************************
15 REM *OEMONSTRATION PROGRAM TO SHOW *
20 REM *SOME OF THE APPLICATIONS FOR  *
25 REM *THE FOLLOWING ROUTINES IN THE *
30 REM *      GRAPHIS PACKAGE:        *
35 REM *CURSOR CONTROL - SYS(29216)   *
40 REM *HORIZONTAL LINE- SYS(28800)   *
45 REM *VERTICAL LINE  - SYS(28832)   *
50 REM *    BOROERS    - SYS(29024)   *
55 REM *BLOCK DISPLAY  - SYS(28896)   *
60 REM *BLOCK REVERSE  - SYS(28960)   *
65 REM *THIS PROGRAM IS JUST PART OF A*
70 REM *HYPOTHETICAL APPLICATION AND  *
75 REM *IS ONLY INTENOEO AS A OEMO    *
80 REM ******************************
130 POKE55,0:POKE56,112
140 REM * LIMIT TOP OF MEMORY
145 POKE51,0:POKE52,112:CLR
150 T=0:GT=0:L=4:LA=18
160 PRINT""
170 REM
180 REM
190 REM * SET UP BORDERS
200 REM
210 REM
214 POKE53281,12:POKE53280,2
215 C=1
220 POKE89,0:POKE90,0:POKE91,38:POKE92,15
225 POKE786,C:SYS(29024)
230 POKE89,0:POKE90,15:POKE91,18:POKE92,10
235 POKE786,C:SYS(29024)
240 POKE89,20:POKE90,15:POKE91,18:POKE92,10
245 POKE786,C:SYS(29024)
250 REM
260 REM
270 REM * SET UP HORIZONTAL ANO
275 REM *     VERTICAL BARS
280 REM
290 REM
295 C=1
300 POKE89,20:POKE90,2:POKE91,9:POKE784,84
305 POKE786,C:SYS(28832)
310 POKE89,25:POKE90,2:POKE91,9:POKE784,84
315 POKE786,C:SYS(28832)
320 POKE89,31:POKE90,2:POKE91,9:POKE784,84
325 POKE786,C:SYS(28832)
330 POKE89,30:POKE90,11:POKE91,7:POKE784,70
335 POKE786,C:SYS(28800)
340 REM
350 REM
360 REM * DISPLAY CHARACTER TO INDICATE
```

108

```
370 REM * ADDRESS INPUT LIMITS
380 REM
390 REM
400 POKE89,2:POKE90,18:POKE91,16:POKE92,5
405 POKE784,100:POKE786,C:SYS(28896)
410 REM
420 REM
430 REM * SET UP SCREEN TEXT USING
431 REM *       CURSOR
435 REM * CONTROL ROUTINE EACH ITEM OF
440 REM * TEXT IS HELD AS THREE ELEMENTS
445 REM * IN A DATA STATEMENT - THEY ARE
450 REM * COLUMN #, LINE #, AND TEXT
460 REM
470 REM
480 DATA4,2,DESCRIPTION
490 DATA21,2,QTY
500 DATA26,2,COST
510 DATA32,2,TOTAL
520 DATA25,12,TOTAL
530 DATA3,16,NAME & ADDRESS
540 DATA23,16,FUNCTION - ?
550 DATA23,18,1 - UPDATE
560 DATA23,20,2 - DELETE
570 DATA23,22,3 - EXIT
580 FORQ=1TO10
590 READA:POKE88,A: REM * COLUMN #
600 READA:POKE87,A: REM * LINE #
610 SYS(29216):     REM * MOVE CURSOR
620 READA$:PRINT"▊"A$: REM * DISPLAY TEXT
630 NEXTQ
640 REM
650 REM
660 REM * SELECT AND RVS FEILD ACTIVE
665 REM * SCREEN FUNCTION FROM MENU
670 REM * ENCLOSED IN BOTTOM LEFT BORDER
680 REM
690 REM
700 REM
710 REM * RVS TO HIGHLIGHT WORD FUNCTION
715 REM *           IN MENU
720 REM
730 POKE89,23:POKE90,16:POKE91,12:POKE92,1
735 POKE786,2:SYS(28960)
740 GETA$:IFA$=""THEN740
750 REM
760 REM * RVS TO REMOVE HIGHLIGHT FROM
765 REM *           'FUNCTION'
770 REM
780 POKE89,23:POKE90,16:POKE91,12:POKE92,1
785 POKE786,1:SYS(28960)
```

```
790 A=VAL(A$):ONAGOTO830,2300,2630
800 GOTO640
810 REM
820 REM
830 REM *'UPDATE FUNCTION DEMO ONLY
840 REM
850 REM
860 REM
870 REM * RVS TO HIGHLIGHT 'UPDATE'
875 REM *          IN MENU
880 REM
890 POKE89,23:POKE90,18:POKE91,10:POKE92,1
895 POKE786,2:SYS(28960)
900 IFL<10THEN1040
910 REM
920 REM * CURSOR CONTROL TO DISPLAY
925 REM *        ERROR MESSAGE
930 REM
940 POKE87,12:POKE85,2:SYS(29216)
950 PRINT"█▓RECORD FULL█"
960 GOTO1880
970 REM
980 REM * INPUT DESCRIPTION
990 REM
1000 REM
1010 REM * HIGHLIGHT 'DESCRIPTION' AND
1015 REM *    PUT CURSOR TO
1020 REM * DISPLAY INPUT ON LINE 'L'.
1030 REM
1040 POKE89,4:POKE90,2:POKE91,11:POKE92,1
1045 POKE786,2:SYS(28960)
1050 POKE87,L:POKE88,4:SYS(29216)
1060 D$=""
1070 GETA$:IFA$=""THEN1070
1080 REM
1090 REM * IF INPUT IS '↑' THEN ABORT TO
1095 REM *    NEXT SECTION AND
1100 REM * REMOVE HIGHLIGHT FROM
1105 REM *      'DESCRIPTION'
1110 REM
1120 IFA$<>"↑"THEN1130
1125 POKE89,4:POKE90,2:POKE91,11:POKE92,1
1126 POKE786,1:SYS(28960):GOTO2000
1130 D$=D$+A$:IFA$=CHR$(13)THENGOTO1180
1140 PRINT"█"A$;:GOTO1070
1150 REM
1160 REM * REMOVE HIGHLIGHT RVS FROM
1165 REM *         'DESCRIPTION'
1170 REM
1180 POKE89,4:POKE90,2:POKE91,11:POKE92,1
1185 POKE786,1:SYS(28960)
```

```
1190 REM
1200 REM * INPUT QUANTITY
1210 REM
1220 REM * RVS FEILD TO HIGHLIGHT 'QTY'
1225 REM *      AND MOVE CURSOR TO
1230 REM * DISPLAY INPUT ON LINE 'L'.
1240 REM
1250 POKE89.21:POKE90.2:POKE91.3:POKE92.1
1255 POKE786.2:SYS(28960)
1260 POKE87.L:POKE88.21:SYS(29216)
1270 Q$=""
1280 GETA$:IFA$=""THEN1280
1290 Q$=Q$+A$:IFA$=CHR$(13)THENGOTO1340
1300 PRINT"▥"A$;:GOTO1280
1310 REM
1320 REM * REMOVE RVS HIGHLIGHT FROM
1325 REM *           'QTY'
1330 REM
1340 POKE89.21:POKE90.2:POKE91.3:POKE92.1
1345 POKE786.1:SYS(28960)
1350 REM
1360 REM * INPUT COST
1370 REM
1380 REM * RVS TO HIGHLIGHT 'COST' AND
1385 REM *   PUT CURSOR ON LINE 'L'
1390 REM * TO DISPLAY INPUT
1400 REM
1410 POKE89.26:POKE90.2:POKE91.4:POKE92.1
1415 POKE786.2:SYS(28960)
1420 POKE87.L:POKE88.26:SYS(29216)
1430 C$=""
1440 GETA$:IFA$=""THEN1440
1450 C$=C$+A$:IFA$=CHR$(13)THENGOTO1500
1460 PRINT"◣"A$;:GOTO1440
1470 REM
1480 REM * REMOVE RVS HIGHLIGHT FROM
1485 REM *           'COST'
1490 REM
1500 POKE89.26:POKE90.2:POKE91.4:POKE92.1
1505 POKE786.1:SYS(28960)
1510 REM
1520 REM * CALCULATE AND DISPLAY TOTAL
1530 REM
1540 REM * RVS TO HIGHLIGHT 'TOTAL'
1550 REM
1560 POKE89.32:POKE90.2:POKE91.5:POKE92.1
1565 POKE786,2:SYS(28960)
1570 T=VAL(Q$)*VAL(C$):GT=GT+T
1580 REM
1590 REM * PUT CURSOR ON LINE 'L' AT
1595 REM *   POSITON TO DISPLAY TOTAL
```

```
1600 REM
1610 POKE87.L:POKE88,32:SYS(29216)
1620 PRINT"█"T
1630 REM
1640 REM * REMOVE RVS HIGHLIGHT FROM
1645 REM *         'TOTAL'
1650 REM
1660 POKE89,32:POKE90,2:POKE91,5:POKE92,1
1665 POKE786,1:SYS(28960)
1670 REM
1680 REM * PROMPT FOR ANOTHER ENTRY
1690 REM
1700 REM * MOVE CURSOR TO LOCATION FOR
1705 REM *    PROMPT AND DISPLAY
1710 REM
1720 POKE87.13:POKE88,2:SYS(29216)
1730 PRINT"██ANOTHER ENTRY - Y OR N█"
1740 GETA$:IFA$=""THEN1740
1750 L=L+1
1760 REM
1770 REM * USE HORIZONTAL BAR OF SPACE
1775 REM *    CHARS TO DELETE PROMPT
1780 REM
1790 POKE89,1:POKE90,13:POKE91,22:POKE784,32
1795 POKE786,0:SYS(28800)
1800 IFA$="N"THEN1880
1810 GOTO900
1820 REM
1830 REM * CALCULATE GRAND TOTAL FOR ALL
1835 REM *           ENTRIES
1840 REM
1850 REM * RVS TO HIGHLIGHT 'TOTAL' AND
1855 REM *    MOVE CURSOR TO
1860 REM * LOCATION TO DISPLAY GRAND
1865 REM *        TOTAL
1870 REM
1880 POKE89,25:POKE90,12:POKE91,5:POKE92,1
1885 POKE786,2:SYS(28960)
1890 POKE87.12:POKE88,32:SYS(29216)
1900 PRINT"█"GT:FORQ=1TO100:NEXTQ
1910 REM
1920 REM * REMOVE RVS 'HIGHLIGHT' FROM
1925 REM *      TOTAL
1930 REM
1940 POKE89,25:POKE90,12:POKE91,5:POKE92,1
1945 POKE786,1:SYS(28960)
1950 REM
1960 REM * INPUT NAME AND ADDRESS
1970 REM
1980 REM * RVS HIGHLIGHT 'NAME AND
1985 REM *      'ADDRESS'
```

112

```
1990 REM
2000 POKE89.3:POKE90.16:POKE91.14:POKE92.1
2005 POKE786.2:SYS(28960)
2010 LA=18
2020 REM
2030 REM * MOVE CURSOR TO START OF NAME
2035 REM *     AND ADDRESS INPUT
2040 REM *     ON LINE 'LA'
2050 REM
2060 POKE87.LA:POKE88.2:SYS(29216)
2070 GETA$:IFA$=""THEN2070
2080 IFA$="↑"THEN2210
2090 IFA$=CHR$(13)THEN2070
2100 PRINT"▉"A$;:AD$=AD$+A$
2110 IFA$="."THENLA=LA+1:GOTO2140
2120 IFA$="."THENGOTO2160
2130 GOTO2070
2140 IFLA>22THENLA=22
2150 GOTO2060
2160 GETA$:IFA$=""THEN2160
2170 IFA$<>CHR$(13)THEN2070
2180 REM
2190 REM * RVS TO REMOVE HIGHLIGHT FROM
2195 REM *     'NAME AND ADDRESS'
2200 REM
2210 POKE89.3:POKE90.16:POKE91.14:POKE92.1
2215 POKE786,1:SYS(28960)
2220 REM
2230 REM * REMOVE HIGHLIGHT FROM FUNCTION
2235 REM *        'UPDATE'
2240 REM * AND RETURN TO FUNCTION SELECT
2250 REM
2260 POKE89.23:POKE90.18:POKE91.10:POKE92.1
2265 POKE786,1:SYS(28960)
2270 GOTO640
2280 REM
2290 REM
2300 REM *'DELETE RECORD FUNCTION
2310 REM
2320 REM
2330 D$="":Q$="":C$="":AD$=""
2340 REM
2350 REM * RVS TO HIGHLIGHT 'DELETE'
2355 REM *     FUNCTION
2360 REM
2370 POKE89.23:POKE90.20:POKE91.10:POKE92.1
2375 POKE786.2:SYS(28960)
2380 T=0:GT=0:L=4:LA=18
2390 REM
2400 REM * DELETE BLOCKS OF TEXT FROM
2405 REM *  SCREEN USING THE ROUTINE
```

```
2410 REM * TO FILL A SCREEN BLOCK WITH
2415 REM *  SPECIFIED CHARACTER
2420 REM * - HERE A SPACE CHARACTER
2430 REM
2440 POKE89,2:POKE90,4:POKE91,18:POKE92,7
2445 POKE784,32:POKE786,0:SYS(28896)
2450 POKE89,21:POKE90,4:POKE91,4:POKE92,7
2455 POKE784,32:POKE786,0:SYS(28896)
2460 POKE89,26:POKE90,4:POKE91,4:POKE92,7
2465 POKE784,32:POKE786,0:SYS(28896)
2470 POKE89,32:POKE90,4:POKE91,6:POKE92,7
2475 POKE784,32:POKE786,0:SYS(28896)
2480 REM
2490 REM * DELETE LINE USING HORIZONTAL
2495 REM *    LINE DRAWING WITH
2500 REM * SPACE CHARACTER
2510 REM
2520 POKE89,32:POKE90,12:POKE91,5:POKE784,32
2525 POKE786,0:SYS(28800)
2530 REM
2540 REM * DISPLAY CHARACTER BLOCK TO
2545 REM *  GIVE ADDRESS INPUT LIMITS
2550 REM
2560 POKE89,2:POKE90,18:POKE91,16:POKE92,5
2565 POKE784,100:POKE786,1:SYS(28896)
2570 REM
2580 REM * RVS TO REMOVE HIGHLIGHT FROM
2585 REM * 'DELETE' AND
2590 REM * RETURN TO FUNCTION SELECT.
2600 REM
2610 POKE89,23:POKE90,20:POKE91,10:POKE92,1
2615 POKE768,1:SYS(28960)
2620 GOTO640
2630 STOP
READY.
```

FINE RESOLUTION PLOTTING

A great drawback in having a display only 40 characters wide and 25 lines deep is the poor definition achievable when displaying data in graphical form. A limitation particularly apparent when trying to display data as bar charts, either vertical or horizontal, on the screen. Fortunately the character set on the Commodore 64 allows this limitation to be overcome. By careful use of specific graphic characters in both normal and reverse field a single character space can be resolved into eight lines in either horizontal or vertical directions. The resolution of a character space thus achieved can take two forms, either a set of eight characters each having a reverse field line of increasing width (from 1/8 character to 8/8 of a character), or a set of characters each with a single thin line but positioned in the 1/8 character increments.

The first of these forms can be used with normal reverse field spaces to create bar charts with resolutions of one part in 320 in the horizontal and 200 in the vertical. The second form is used where a thin, fine incremented line is required, such as the example program in this section, plotting a graph of a curve. For each of these two forms of fine resolution plotting there are two sets of characters, one set for vertical and the other for horizontal plotting.

HORIZONTAL BAR LEFT TO RIGHT

This routine draws a horizontal bar of reverse field spaces from left to right in fine resolution. The left hand co-ordinates are determined by two variables, column number and line number. The length of the bar is set by a variable and has a resolution of 1 in 135, the minimum length of the bar is 0 and the maximum is 135, these values are the number of 1/8 ths of a normal character space. Attempts to draw a bar longer than 135 will give an erroneous display. The variable locations required by the routine are as follows:

89 ... column number of start of fine resolution bar plot (left end).
90 ... line number of start of fine resolution plot.
91 ... length of fine resolution bar plot in 1/8 ths of a character space.
786 ... colour of bar.
The routine is called by SYS(29258).

HORIZONTAL BAR RIGHT TO LEFT

This routine draws a horizontal bar of reverse field spaces from right to left in fine resolution. The right hand co-ordinates are determined by two variables, column number and line number. The length of the bar is set by a variable and has a resolution of 1 in 135, the minimum length of the bar is 0 and the maximum is 135, these values are the number of 1/8 ths of a normal character space. Attempts to draw a bar longer than 135 will give an erroneous display. The variable locations required by the routine are as follows:

89 ... column number of start of fine resolution bar plot (right end).

90 ... line number of start of fine resolution plot.

91 ... length of fine resolution bar plot in 1/8 ths of a character space.

786 ... colour of bar.

The routine is called by SYS(29328).

VERTICAL BAR BOTTOM TO TOP

This routine draws a vertical bar of reverse field spaces from bottom to top in fine resolution. The bottom co-ordinates are determined by two variables, column number and line number. The length of the bar is set by a variable and has a resolution of 1 in 135, the minimum length of the bar is 0 and the maximum is 135, these values are the number of 1/8 ths of a normal character space. Attempts to draw a bar longer than 135 will give an erroneous display. The variable locations required by the routine are as follows:

89 ... column number of start of fine resolution bar plot (bottom end).

90 ... line number of start of fine resolution plot.

91 ... length of fine resolution bar plot in 1/8 ths of a character space.

786 ... colour of bar.

The routine is called by SYS(29408).

VERTICAL BAR TOP TO BOTTOM

This routine draws a vertical bar of reverse field spaces from top to bottom in fine resolution. The top hand co-ordinates are determined by two variables, column number and line number. The length of the bar is set by a variable and has a resolution of 1 in 135, the minimum length of the bar is 0 and the maximum is 135, these values are the number of 1/8'ths of a normal character space. Attempts to draw a bar longer than 135 will give an erroneous display. The variable locations required by the routine are as follows:

89 ... column number of start of fine resolution bar plot (top end).
90 ... line number of start of fine resolution plot.
91 ... length of fine resolution bar plot in 1/8'ths of a character space.
786 ... colour of bar.

The routine is called by SYS(29503).

HORIZONTAL LINE LEFT TO RIGHT

This routine draws a bar of space characters from left to right terminated by a fine resolution vertical line one eighth of a character wide. The left hand co-ordinates are determined by two variables, column number and line number. The length of the bar is set by a variable and has a resolution of 1 in 135, the minimum length of the bar is 0 and the maximum is 135, these values are the number of 1/8 ths of a normal character space. Attempts to draw a bar longer than 135 will give an erroneous display. The variable locations required by the routine are as follows:

89 ... column number of start of fine resolution bar plot (left end).
90 ... line number of start of fine resolution plot.
91 ... length of fine resolution bar plot in 1/8 ths of a character space.
786 ... colour of bar.

The routine is called by SYS(29600).

HORIZONTAL LINE RIGHT TO LEFT

This routine draws a bar of space characters from right to left terminated by a fine resolution vertical line one eighth of a character wide. The right hand co-ordinates are determined by two variables, column number and line number. The length of the bar is set by a variable and has a resolution of 1 in 135, the minimum length of the bar is 0 and the maximum is 135, these values are the number of 1/8'ths of a normal character space. Attempts to draw a bar longer than 135 will give an erroneous display. The variable locations required by the routine are as follows:

89 ... column number of start of fine resolution bar plot (right end).

90 ... line number of start of fine resolution plot.

91 ... length of fine resolution bar plot in 1/8 ths of a character space.

786 ... colour of bar.

The routine is called by SYS(29680).

VERTICAL LINE TOP TO BOTTOM

This routine draws a bar of space characters from left to right terminated by a fine resolution horizontal line one eighth of a character wide. The top co-ordinates are determined by two variables, column number and line number. The length of the bar is set by a variable and has a resolution of 1 in 135, the minimum length of the bar is 0 and the maximum is 135, these values are the number of 1/8 ths of a normal character space. Attempts to draw a bar longer than 135 will give an erroneous display. The variable locations required by the routine are as follows:

89 ... column number of start of fine resolution bar plot (top end).

90 ... line number of start of fine resolution plot.

91 ... length of fine resolution bar plot in 1/8 ths of a character space.

786 ... colour of bar.

The routine is called by SYS(29856).

VERTICAL LINE BOTTOM TO TOP

This routine draws a bar of space characters from bottom to top terminated by a fine resolution horizontal line one eighth of a character wide. The bottom co-ordinates are determined by two variables, column number and line number. The length of the bar is set by a variable and has a resolution of 1 in 135, the minimum length of the bar is 0 and the maximum is 135, these values are the number of 1/8 ths of a normal character space. Attempts to draw a bar longer than 135 will give an erroneous display. The variable locations required by the routine are as follows:

89 ... column number of start of fine resolution bar plot (bottom end).

90 ... line number of start of fine resolution plot.

91 ... length of fine resolution bar plot in 1/8 ths of a character space.

786 ... colour of bar.

The routine is called by SYS(29760).

```
10 REM
20 REM ******************************
30 REM *ROUTINE TO CALCULATE SCREEN  *
40 REM *ADDRESS FROM VALUES FOR      *
50 REM *    COLUMN       IN   89     *
60 REM *    LINE         IN   90     *
70 REM *ADDRESS RETURNED IN 91 AND 92*
80 REM *THIS ROUTINE IS ONLY CALLED  *
90 REM *BY OTHER ROUTINES IN THE     *
100 REM*PACKAGE,DO NOT USE BY ITSELF.*
110 REM*ERROR FLAG IN LOCATION 785   *
120 REM******************************
130 I=28672:T=0
140 READ A
150 IF A=-1 THEN 190
160 T=T+A:POKEI,A
170 I=I+1
180 GOTO 140
190 PRINT":ROUTINE 1:"
200 IF T=11973 THEN PRINT"XENTERED O.K":GOTO 1000
210 PRINT"XENTERED INCORRECTLY"
220 END
230 DATA169,0,141,17,3,165,89
240 DATA48,66,201,40,176,74,165
250 DATA90,48,82,201,25,176,90
260 DATA169,0,133,87,133,88,133
270 DATA94,133,95,165,90,240,15
280 DATA170,24,165,87,105,40,133
290 DATA87,144,2,230,88,202,208
300 DATA242,24,165,87,101,89,133
310 DATA87,165,88,105,4,133,88
320 DATA24,165,87,133,94,165,88
330 DATA105,212,133,95,96,169,1
340 DATA141,17,3,169,0,133,89
350 DATA76,7,112,169,2,141,17
360 DATA3,169,39,133,89,76,7
370 DATA112,169,3,141,17,3,169
380 DATA0,133,90,76,7,112,169
390 DATA4,141,17,3,169,24,133
400 DATA90,76,7,112,-1
1000 REM
1010 REM******************************
1020 REM*DRAW A BORDER OF ANY SIZE,ANY*
1030 REM*LOCATION, AND ANY COLOUR. TOP*
1040 REM*LEFT COORDINATES ARE:        *
1050 REM*    COLUMN       IN   89     *
1060 REM*    LINE         IN   90     *
1070 REM*    WIDTH        IN   91     *
1080 REM*    HEIGHT       IN   92     *
1090 REM* COLOUR IS STORED IN 786.    *
1100 REM*ROUTINE CALLED BY SYS(29024) *
```

```
1110 REM*****************************
1120 I=29024:T=0
1130 READ A
1140 IF A=-1 THEN1180
1150 POKE I,A:T=T+A
1160 I=I+1
1170 GOTO1130
1180 PRINT"ROUTINE 2:"
1190 IF T=20557 THEN PRINT"ENTERED O.K.":GOTO2000
1200 PRINT"ENTERED INCORRECTLY"
1210 END
1220 DATA72,152,72,138,72,32,0
1230 DATA112,165,87,133,89,165,88
1240 DATA133,90,169,100,141,16,3
1250 DATA32,212,113,24,165,87,105
1260 DATA41,133,87,133,89,165,88
1270 DATA105,0,133,88,133,90,169
1280 DATA101,141,16,3,32,239,113
1290 DATA24,165,89,101,91,133,87
1300 DATA198,87,165,90,105,0,133
1310 DATA88,169,103,141,16,3,32
1320 DATA239,113,165,89,133,87,165
1330 DATA90,133,88,166,92,198,87
1340 DATA202,202,24,165,87,105,40
1350 DATA133,87,165,88,105,0,133
1360 DATA88,202,208,240,169,99,141
1370 DATA16,3,32,212,113,104,170
1380 DATA104,168,104,96,164,91,165
1390 DATA87,133,94,165,88,24,105
1400 DATA212,133,95,173,16,3,145
1410 DATA87,173,18,3,145,94,136
1420 DATA208,243,96,160,0,166,92
1430 DATA202,202,173,16,3,145,87
1440 DATA24,165,87,133,94,165,88
1450 DATA105,212,133,95,173,18,3
1460 DATA145,94,24,165,87,105,40
1470 DATA133,87,165,88,105,0,133
1480 DATA88,202,208,219,96,-1
2000 REM
2010 REM*****************************
2020 REM*ROUTINE TO PLACE THE CURSOR  *
2030 REM*AT A LOCATION ON THE SCREEN  *
2040 REM*WHOSE COORDINATES ARE STORED *
2050 REM*IN   87=LINE AND 88=COLUMN.  *
2060 REM*ROUTINE  CALLED BY SYS(29216)*
2070 REM*****************************
2080 I=29216:T=0
2090 READ A
2100 IF A=-1 THEN2140
2110 POKE I,A:T=T+A
2120 I=I+1
```

```
2130 GOTO2090
2140 PRINT"⊐ROUTINE 3:"
2150 IF T=5163 THEN PRINT"█ENTERED O.K.":GOTO3000
2160 PRINT"█ENTERED INCORRECTLY"
2170 END
2180 DATA72,152,72,138,72,169,19
2190 DATA32,22,231,165,87,240,9
2200 DATA169,17,32,22,231,198,87
2210 DATA208,247,165,88,240,9,169
2220 DATA29,32,22,231,198,88,208
2230 DATA247,104,170,104,168,104,96,-1
3000 REM
3010 REM*********************************
3020 REM*DRAW HORIZONTAL BAR - LEFT TO*
3030 REM*RIGHT. RESOLUTION 135 X 25.  *
3040 REM*COLUMN # IN 89, LINE # IN 90.*
3050 REM*LENGTH    IN 91. COLOUR IN 786*
3060 REM*ROUTINE CALLED BY SYS(29258) *
3070 REM*********************************
3080 I=29258:T=0
3090 READ A
3100 IF A=-1 THEN3140
3110 POKE I,A:T=T+A
3120 I=I+1
3130 GOTO3090
3140 PRINT"⊐ROUTINE 4:"
3150 IF T=7891 THEN PRINT"█ENTERED O.K.":GOTO4000
3160 PRINT"█ENTERED INCORRECTLY"
3170 END
3180 DATA72,152,72,138,72,32,0
3190 DATA112,160,0,165,91,201,8
3200 DATA48,27,233,8,133,91,169
3210 DATA160,145,87,173,18,3,145
3220 DATA94,24,230,87,230,94,144
3230 DATA4,230,88,230,95,76,84
3240 DATA114,170,189,134,114,145,87
3250 DATA173,18,3,145,94,104,170
3260 DATA104,168,104,96,32,101,116
3270 DATA117,97,246,234,231,160,-1
4000 REM
4010 REM*********************************
4020 REM*DRAW HORIZONTAL BAR - RIGHT  *
4030 REM*TO LEFT. RESOLUTION 135 X 25.*
4040 REM*COLUMN # IN 89,              *
4050 REM*LINE   # IN 90,              *
4060 REM*LENGTH    IN 91,             *
4070 REM*COLOUR    IN 786.            *
4080 REM*ROUTINE CALLED BY SYS(29328) *
4090 REM*********************************
4100 I=29328:T=0
4110 READ A
```

```
4120 IF A=-1 THEN4160
4130 POKE I,A:T=T+A
4140 I=I+1
4150 GOTO4110
4160 PRINT"⊃ROUTINE 5:"
4170 IF T=8095 THEN PRINT"▧▧ENTERED O.K.":GOTO5000
4180 PRINT"▧▧ENTERED INCORRECTLY"
4190 END
4200 DATA72,152,72,138,72,32,0
4210 DATA112,160,0,165,91,201,8
4220 DATA48,27,233,8,133,91,169
4230 DATA160,145,87,173,18,3,145
4240 DATA94,56,198,87,198,94,176
4250 DATA4,198,88,198,95,76,154
4260 DATA114,170,189,204,114,145,87
4270 DATA173,18,3,145,94,104,170
4280 DATA104,168,104,96,32,103,106
4290 DATA118,225,245,244,229,160,-1
5000 REM
5010 REM*******************************
5020 REM*DRAW VERTICAL 8AR - BOTTOM TO*
5030 REM*TOP. RESOLUTION 135 X 40.    *
5040 REM*COLUMN # IN 89.              *
5050 REM*LINE   # IN 90.              *
5060 REM*LENGTH   IN 91.              *
5070 REM*COLOUR   IN 786.             *
5080 REM*ROUTINE CALLED BY SYS(29408) *
5090 REM*******************************
5100 I=29408:T=0
5110 READ A
5120 IF A=-1 THEN5160
5130 POKE I,A:T=T+A
5140 I=I+1
5150 GOTO5110
5160 PRINT"⊃ROUTINE 6:"
5170 IF T=9031 THEN PRINT"▧▧ENTERED O.K.":GOTO6000
5180 PRINT"▧▧ENTERED INCORRECTLY"
5190 END
5200 DATA72,152,72,138,72,32,0
5210 DATA112,160,0,165,91,201,8
5220 DATA48,37,233,8,133,91,169
5230 DATA160,145,87,173,18,3,145
5240 DATA94,24,165,87,233,39,133
5250 DATA87,133,94,165,88,233,0
5260 DATA133,88,24,105,212,133,95
5270 DATA165,88,208,213,170,189,38
5280 DATA115,145,87,173,18,3,145
5290 DATA94,104,170,104,168,104,96
5300 DATA32,100,111,121,98,248,247
5310 DATA227,160,0,-1
6000 REM
```

```
6010 REM******************************
6020 REM*DRAW VERTICAL BAR - TOP TO   *
6030 REM*BOTTOM. RESOLUTION 135 X 25. *
6040 REM*COLUMN # IN 89.              *
6050 REM*LINE   # IN 90,              *
6060 REM*LENGTH   IN 91.              *
6070 REM*COLOUR   IN 786.             *
6080 REM*ROUTINE CALLED BY SYS(29503) *
6090 REM******************************
6100 I=29503:T=0
6110 READ A
6120 IF A=-1 THEN6160
6130 POKE I,A:T=T+A
6140 I=I+1
6150 GOTO6110
6160 PRINT"⊐ROUTINE 7:"
6170 IF T=8999 THEN PRINT"XXENTERED O.K.":GOTO7000
6180 PRINT"XXENTERED INCORRECTLY"
6190 END
6200 DATA72,152,72,138,72,32,0
6210 DATA112,160,0,165,91,201,8
6220 DATA48,37,233,8,133,91,169
6230 DATA160,145,87,173,18,3,145
6240 DATA94,24,165,87,105,40,133
6250 DATA87,133,94,165,88,105,0
6260 DATA133,88,24,105,212,133,95
6270 DATA165,88,208,213,170,189,133
6280 DATA115,145,87,173,18,3,145
6290 DATA94,104,170,104,168,104,96
6300 DATA32,99,119,120,226,249,239
6310 DATA228,160,0,-1
7000 REM
7010 REM******************************
7020 REM*DRAW HORIZONTAL BAR - LEFT TO*
7030 REM*RIGHT OF SPACES TERMINATED IN*
7040 REM*A THIN LINE. RESOLUTION      *
7050 REM* 135 X 25.                   *
7060 REM*COLUMN # IN 89.              *
7070 REM*LINE   # IN 90,              *
7080 REM*LENGTH   IN 91.              *
7090 REM*COLOUR   IN 786.             *
7100 REM*ROUTINE CALLED BY SYS(29600) *
7110 REM******************************
7120 I=29600:T=0
7130 READ A
7140 IF A=-1 THEN7180
7150 POKE I,A:T=T+A
7160 I=I+1
7170 GOTO7130
7180 PRINT"⊐ROUTINE 8:"
7190 IF T=7251 THEN PRINT"XXENTERED O.K.":GOTO8000
```

```
7200 PRINT"XXENTERED INCORRECTLY"
7210 END
7220 DATA72,152,72,138,72,32,0
7230 DATA112,160,0,165,91,201,7
7240 DATA48,27,233,7,133,91,169
7250 DATA32,145,87,173,18,3,145
7260 DATA94,24,230,87,230,94,144
7270 DATA4,230,88,230,95,76,170
7280 DATA115,170,189,220,115,145,87
7290 DATA173,18,3,145,94,104,170
7300 DATA104,168,104,96,101,84,71
7310 DATA66,72,89,103,32,32,-1
8000 REM
8010 REM*******************************
8020 REM*DRAW HORIZONTAL BAR - RIGHT  *
8030 REM*TO LEFT OF SPACES TERMINATED *
8040 REM*IN A THIN LINE. RESOLUTION   *
8050 REM*135 X 25.                    *
8060 REM*COLUMN # IN 89.              *
8070 REM*LINE   # IN 90.              *
8080 REM*LENGTH   IN 91.              *
8090 REM*COLOUR   IN 786.             *
8100 REM*ROUTINE CALLED BY SYS(29680) *
8110 REM*******************************
8120 I=29680:T=0
8130 READ A
8140 IF A=-1 THEN8180
8150 POKE I,A:T=T+A
8160 I=I+1
8170 GOTO8130
8180 PRINT"XROUTINE 9:"
8190 IF T=7092 THEN PRINT"XXENTERED O.K.":GOTO9000
8200 PRINT"XXENTERED INCORRECTLY"
8210 END
8220 DATA72,152,72,138,72,32,0
8230 DATA112,160,0,165,91,201,7
8240 DATA48,27,233,7,133,91,169
8250 DATA32,145,87,173,18,3,145
8260 DATA94,56,198,87,198,94,176
8270 DATA4,198,88,198,95,76,250
8280 DATA115,170,189,44,116,145,87
8290 DATA173,18,3,145,94,104,170
8300 DATA104,168,104,96,103,89,72
8310 DATA66,71,84,101,32,32,-1
9000 REM
9010 REM*******************************
9020 REM*DRAW VERTICAL BAR - BOTTOM TO*
9030 REM*TOP OF SPACES TERMINATED IN A*
9040 REM*THIN LINE.RESOLUTION 135 X 40*
9050 REM*COLUMN # IN 89.              *
9060 REM*LINE   # IN 90,              *
```

```
9070 REM*LENGTH    IN 91.            *
9080 REM*COLOUR    IN 786.           *
9090 REM*ROUTINE CALLED BY SYS(29760) *
9100 REM*******************************
9110 I=29760:T=0
9120 READ A
9130 IF A=-1 THEN9170
9140 POKE I,A:T=T+A
9150 I=I+1
9160 GOTO9120
9170 PRINT"[]ROUTINE 10:"
9180 IF T=8307 THEN PRINT"[]ENTERED O.K.":GOTO10000
9190 PRINT"[]ENTERED INCORRECTLY"
9200 END
9210 DATA72,152,72,138,72,32,0
9220 DATA112,160,0,165,91,201,8
9230 DATA48,37,233,8,133,91,169
9240 DATA32,145,87,173,18,3,145
9250 DATA94,24,165,87,233,39,133
9260 DATA87,133,94,165,88,233,0
9270 DATA133,88,24,105,212,133,95
9280 DATA165,88,208,213,170,189,134
9290 DATA116,145,87,173,18,3,145
9300 DATA94,104,170,104,168,104,96
9310 DATA100,82,70,64,67,68,69
9320 DATA99,32,0,-1
10000 REM
10010 REM******************************
10020 REM*DRAW VERTICAL BAR - TOP TO  *
10030 REM*BOTTOM OF SPACES TERMINATED *
10040 REM*IN A THIN LINE. RESOLUTION  *
10050 REM* 135 X 40.                  *
10060 REM*COLUMN # IN 89.             *
10070 REM*LINE   # IN 90.             *
10080 REM*LENGTH   IN 91,             *
10090 REM*COLOUR   IN 786.            *
10100 REM*ROUTINE CALLED BY SYS(29856) *
10110 REM******************************
10120 I=29856:T=0
10130 READ A
10140 IF A=-1 THEN10180
10150 POKE I,A:T=T+A
10160 I=I+1
10170 GOTO10130
10180 PRINT"[]ROUTINE 11:"
10190 IF T=8180 THEN PRINT"[]ENTERED O.K.":GOTO11000
10200 PRINT"[]ENTERED INCORRECTLY"
10210 END
10220 DATA72,152,72,138,72,32,0
10230 DATA112,160,0,165,91,201,8
10240 DATA48,37,233,8,133,91,169
```

126

```
10250 DATA32,145,87,173,18,3,145
10260 DATA94,24,165,87,105,40,133
10270 DATA87,133,94,165,88,105,0
10280 DATA133,88,24,105,212,133,95
10290 DATA165,88,208,213,170,189,230
10300 DATA116,145,87,173,18,3,145
10310 DATA94,104,170,104,168,104,96
10320 DATA99,69,68,67,64,70,82
10330 DATA100,32,32,0,-1
11000 REM
11010 REM
11020 REM*****************************
11030 REM*DRAW A HORIZONTAL BAR OF ANY *
11040 REM*64 CHARACTER.CHARACTER STORED*
11050 REM*IN 784, COLUMN START IN 89,  *
11060 REM*LINE START IN 90, LENGTH OF  *
11070 REM*LINE IN 91, COLOUR IN 786.   *
11080 REM*ROUTINE CALLED BY SYS(28800) *
11090 REM*****************************
11100 I=28800:T=0
11110 READ A
11120 IF A=-1 THEN11160
11130 POKE I,A:T=T+A
11140 I=I+1
11150 GOTO11110
11160 PRINT"ROUTINE 12:"
11170 IF T=3095 THEN PRINT"ENTERED O.K":GOTO11250
11180 PRINT"ENTERED INCORRECTLY"
11190 END
11200 DATA72,152,72,138,72,32,0
11210 DATA112,164,91,173,16,3,145
11220 DATA87,173,18,3,145,94,136
11230 DATA208,243,104,170,104,168,104
11240 DATA96,-1
11250 POKE51,0:POKE52,112
11260 POKE55,0:POKE56,112:CLR:NEW
READY.
```

FINE RESOLUTION EMULATOR

The loader on pp. 120-7 should be loaded and run once before using this program.

DESCRIPTION

This program emulates the display of an analog edge meter. To move the needle use the 4 and 6 keys respectively. The program calls the following Display Management routines:

CURSOR CONTROL – SYS (29216)
FINE RES HORIZONTAL – SYS (29600)
BORDER – SYS (29024)
HORIZONTAL LINE – SYS (28800)

PROGRAM STRUCTURE

The following is a description of lines of particular interest in the Basic program:

120-135	: Limit top of memory and clear pointers.
190-260	: Uses the cursor control routine to display the text.
320-345	: Draws a border around the meter display.
370-430	: Test for key press and if legal key press, move the meter needle in the correct direction.
460-461	: Erases the previous meter needle display.
470-471	: Displays the new meter needle position.

```
10 REM *******************************
20 REM *THIS PRG EMULATES THE DISPLAY*
30 REM *OF AN ANAOLG EDGE METER, USE *
35 REM *     KEYS 4 & 6              *
40 REM *TO MOVE METER NEEDLE UP OR   *
45 REM *         DOWN.              *
50 REM *ROUTINE USES THE FOLLOWING   *
55 REM *    MACHINE CODE             *
60 REM *     ROUTINES:              *
70 REM * SYS(29216) - CURSOR CONTROL *
80 REM * SYS(29600) - FINE-RES       *
85 REM *          HORIZONTAL        *
90 REM * SYS(29024) - BORDER         *
100 REM* SYS(28800) - HORIZONTAL LINE*
110 REM*******************************
120 POKE55,0:POKE56.112
130 REM LIMIT TOP OF MEMORY
135 POKE 51.0:POKE52,112:CLR
140 PRINT"◻"
145 REM
150 REM *DRAW SCALE IN INCREMENTS OF 10
151 REM *   FINE RES LINES
154 POKE53281.2
155 REM *   FINE RES LINES
160 FORQ=130TO0 STEP -10
170 POKE 89.10:POKE90,10:POKE91.Q:POKE786.1
175 SYS(29600)
180 NEXTQ
183 REM
185 REM *DISPLAY TEXT USING CURSOR
186 REM *     CONTROL
187 REM
190 DATA 6,11."◼◼ANALOG EDGE METER◼"
200 DATA 8.9,0
210 DATA 8.28,130
220 DATA 20.7,"◼DIGITAL VALUE ON METER◼"
230 FORQ=1TO4
240 READA:POKE87.A:READA:POKE88.A:SYS(29216)
250 READA$:PRINT"◻"A$
260 NEXTQ
270 REM
280 REM *DRAW BORDERS AROUND METER
285 REM *          DISPLAY
290 REM
320 POKE89.8:POKE90.9:POKE91.21:POKE92,4
325 POKE786,7:SYS(29024)
340 POKE89,6:POKE90.5:POKE91.25:POKE92,9
345 POKE786,7:SYS(29024)
350 REM
354 REM *SEE WHICH KEY PRESSED (LOC 197)
355 REM *  IF KEY = 4 (197=11)
```

```
356 REM *THEN DISPLAY VALUE DECREASED.
357 REM *  IF KEY = 6 (197=19)
358 REM *THEN DISPLAY VALUE INCREASED.
360 REM
370 A=0
380 X=PEEK(197)
390 IFX=19THEN A=A+1:GOTO420
400 IFX=11THEN A=A-1:GOTO420
410 GOTO380
420 IFA<0THENA=0
430 IFA>130THENA=130
434 REM
435 REM *LOCATE CURSOR TO DISPLAY
436 REM *  DIGITAL EQUIVALENT
437 REM
440 POKE87,20:POKE88,30:SYS(29216)
450 PRINT"    ▮▮▮▮▮";A
454 REM
455 REM *ERASE PREVIOUS METER NEEDLE
456 REM *      DISPLAY
457 REM
460 POKE89,10:POKE90,11:POKE91,18:POKE784,32
461 POKE786,2:SYS(28800)
464 REM
465 REM *DISPLAY NEW NEEDLE POSITION
466 REM
470 POKE89,10:POKE90,11:POKE91,A:POKE786,1
471 SYS(29600)
480 GOTO380
READY.
```

GRAPH FUNCTION

The loader on pp. 120-7 should be loaded and run once before using this program.

DESCRIPTION

This program plots the graph of a function and uses the following routines from the Display Management section:

DRAW BORDER – SYS (29024)
CURSOR CONTROL – SYS (29216)
VERTICAL FINE RES LINE – (29760)

PROGRAM STRUCTURE

The following lines are of particular interest in the program:

90-105	:	Limits the top of memory and clears the pointers.
210-215	:	Draws a border around the display.
250-260	:	Displays the text using the cursor control routine.
310-340	:	Plots the graph of the function, ensuring that there are no negative numbers and that the maximum limit for the values is 135.

```
10 REM ********************************
20 REM *USING FINE RESOLUTION LINES   *
25 REM *       THIS PROGRAM           *
30 REM *PLOTS THE GRAPH OF A FUNCTION.*
40 REM *PROGRAM USES THE FOLLOWING    *
45 REM *MACHINE CODE ROUTINES:        *
50 REM *   SYS(29024) - DRAW BORDER   *
60 REM *   SYS(29760) - VERTICAL      *
65 REM *   FINE RES LINE              *
70 REM *   SYS(29216) - CURSOR CONTROL *
80 REM ********************************
90 POKE55,0:POKE56,112
100 REM LIMIT TOP OF MEMORY
105 POKE51,0:POKE52,112:CLR
110 PRINT"⊐"
120 REM
130 REM *DRAW VERTICAL SCALE IN
135 REM *INCREMENTS OF 10 FINE RES LINES
140 REM
145 POKE53281,12
150 FORQ=110TO0STEP-10
160 POKE89,0:POKE90,20:POKE91,Q:POKE786,1
165 SYS(29760)
170 NEXTQ
180 REM
190 REM *DRAW BORDER AROUND DISPLAY
200 REM
210 POKE89,0:POKE90,6:POKE91,38:POKE92,16
215 POKE786,6:SYS(29024)
220 REM
230 REM *DISPLAY TEXT USING CURSOR
235 REM *     CONTROL
240 REM
250 POKE87,7:POKE88,9:SYS(29216)
260 PRINT"    GRAPH OF SINE FUNCTION "
270 REM
280 REM *PLOT GRAPH OF FUNCTION. NOTE
285 REM *  OFFSET GIVEN TO FUNCTION
290 REM *TO ENSURE NO NEGATIVE VALUES
295 REM *  FOR PLOT OR VALUES OVER 135
300 REM
310 FORQ=2TO37
320 A=INT(SIN((Q-2)/5.5)*40)+50
330 POKE89,Q:POKE90,20:POKE91,A:POKE786,2
335 SYS(29760)
340 NEXTQ
350 GETA$:IFA$=""THEN350
355 REM PRESS ANY KEY TO END
360 END
READY.
```

BAR CHART 1

The loader on pp. 120-7 should be loaded and run once before using this program.

DESCRIPTION

This program draws a bar chart of up to 31 values with positive numbers. The Machine Code routines that are called from this Basic program are:
DRAW BORDER – SYS (29024)
CURSOR CONTROL – SYS (29216)
PLOT VERTICAL FINE BAR – SYS (29408)
The REM statements are not needed as they are purely informative. Therefore the program starts at line 100, which limits the memory and clears the pointers. At line 100 P% is set to equal 31 integer numbers. Lines 150-160 are the data for the bar chart. Lines 200-230 read the data into the array P%. Line 240 clears the screen, a sensible statement before attempting to display anything. At line 255 the screen colour is set to grey.

The bar chart is set up in lines 310-340. The data (A) is read and POKEd into locations 87 and 88, then the cursor control routine is called to position the cursor for the next to be displayed. Lines 380 and 385 draw the border around the limits of the bar chart. The POKE's to 89 and 90 set the start of the column at the top left corner and at the screen line number of border start. The POKE's to 91 and 92 define the width and the height of the number. On line 385, location 786 holds the character colour (in this case blue).

Lines 420-460 simply go through a loop 1 to 32 and calculate the plotting scale, the highest value is returned in B and on line 500 it is printed. Lines 540-590 display the barchart using the Vertical barplot routine. The variable 'Q' holds the value of each bar as it passes through the loop and calls the barplot routine to place it on the screen. The colour of each bar is determined by C in line 565, but there is a check in line 566 to make sure that the colour of the bar is not the same as the background or previous bar. Line 60 waits for a key press after the bar chart has been displayed, and then ends the program.

```
10 REM ********************************
20 REM *A SINGLE DATA SET OF UP TO 31 *
25 REM *VALUES ALL +VE NUMBERS        *
30 REM *ARE DISPLAYED BY THIS ROUTINE *
35 REM *AS A BARCHART.                *
40 REM *MACHINE CODE ROUTINES USED    *
45 REM *          ARE:                *
50 REM *  SYS(29024) - DRAW BORDER    *
60 REM *  SYS(29216) - CURSOR CONTROL *
70 REM *  SYS(29408) - PLOT VERTICAL  *
75 REM *   FINE RES BAR               *
80 REM ********************************
90 REM * LIMIT TOP OF MEMORY
100 POKE 55,0:POKE56,112:POKE51,0:POKE52,112:CLR
110 DIMP%(31)
120 REM
130 REM *DATA TO BE DISPLAYED
140 REM
150 DATA 2,5,6,3,9,8,10,15,11,17,24,22,18
155 DATA 16,17,18,19,20,21,18
160 DATA 25,23,28,26,20,13,5,21,28,26,35
170 REM
180 REM *PUT DATA INTO ARRAY P%
190 REM
200 FORQ=1TO31
210 READA
220 P%(Q)=A
230 NEXTQ
240 PRINT""
250 REM
255 POKE53281,12
260 REM *DISPLAY TEXT USING CURSOR
265 REM *         CONTROL
270 REM
280 DATA2,14,"   EXAMPLE BAR CHART "
290 DATA21,2,0
300 DATA23,7,"0..........................31"
310 FORQ=1TO3
320 READA:POKE87,A:READA:POKE88,A:SYS(29216)
330 READA$:PRINT""A$
340 NEXTQ
350 REM
360 REM *DRAW BORDER AROUND DISPLAY
365 REM *        AREA
370 REM
380 POKE89,5:POKE90,3:POKE91,33:POKE92,20
385 POKE786,6:SYS(29024)
390 REM
400 REM
410 REM *DEFINE SCALE FOR PLOTTING
420 B=0
```

```
430 FORQ=1TO31
440 A=P%(Q)
450 IFA>BTHENB=A
460 NEXTQ
470 REM
480 REM *PRINT MAXIMUM DISPLAYED VALUE
490 REM
500 PRINT"■■■■■■■"B
510 REM
520 REM *DISPLAY BARCHART USING VERTICAL
525 REM *    BARPLOT ROUTINE
530 REM
540 A=B/135
550 FORQ=1TO31
560 AS=INT(P%(Q)/A)
565 C=INT(RND(1)*8)+1
566 IFC+240=PEEK(53281)ORC=CCTHEN565
570 POKE90,21:POKE89,Q+6:POKE91,AS:POKE786,C
580 SYS(29408):CC=C
590 NEXTQ
600 GETA$:IFA$=""THEN 600
605 REM **WAIT TILL KEY PRESSED TO END**
READY.
```

135

BAR CHART 2

The loader on pp. 120-7 should be loaded and run once before using this program.

DESCRIPTION

This program displays a bar chart of two sets of data, both having up to 15 numbers where all 15 values are positive. The Machine Code routines used in this program are:
DRAW BORDER – SYS (29024)
CURSOR CONTROL – SYS (29216)
PLOT VERTICAL FINE RES BAR – SYS (29408)
PLOT VERTICAL FINE RES LINE – SYS (29760)
Line 100 limits the memory and clears the pointers. Line 110 sets up an array of integer values in P%. This program uses exactly the same technique as the first barchart, the only differences being the use of the vertical fine res line. The fine res barchart is displayed using lines 600-640.

The third program displaying a bar chart uses a single set of data of up to 30 values, the values can be negative or positive. This program uses the following Machine Code routines:
DRAW BORDER – SYS (29024)
CURSOR CONTROL – SYS (29216)
PLOT VERTICAL FINE RES BAR – SYS (29408)
PLOT VERTICAL FINE RES LINE – SYS (29503)

PROGRAM STRUCTURE

The lines of particular interest that are different from the other barchart programs are:
Lines 615-640 : This routine displays the negative bars.
The positive bars are displayed in the upper half of the border and the negative bars are displayed in the lower half.

```
10 REM *********************************
20 REM *TWO DATA SETS EACH OF UP TO 15*
25 REM *VALUES ALL +VE NUMBERS        *
30 REM *ARE DISPLAYED BY THIS ROUTINE *
35 REM *AS A BARCHART.                *
40 REM *MACHINE CODE ROUTINES USED    *
45 REM *ARE:                          *
50 REM *   SYS(29024) - DRAW BORDER   *
60 REM *   SYS(29216) - CURSOR CONTROL*
70 REM *   SYS(29408) - PLOT VERTICAL *
74 REM *   FINE RES BAR               *
75 REM *   SYS(29760) - PLOT VERTICAL *
76 REM *   FINE RES LINE              *
80 REM *********************************
90 REM LIMIT TOP OF MEMORY
100 POKE 55,0:POKE56,112:POKE51,0:POKE52,112:CLR
110 DIMP%(31)
120 REM
130 REM *DATA TO BE DISPLAYED
140 REM
150 DATA 2,1,4,2,6,3,8,4,10,5,12,6,14,7,16
155 DATA 8,18,9,20,10
160 DATA 22,11,24,12,26,13,28,14,30,15
170 FORQ=1TO30
180 READA
190 P%(Q)=A
200 NEXTQ
210 REM
220 REM *DISPLAY TEXT USING CURSOR
225 REM *       CONTROL
230 REM
240 PRINT""
245 POKE53281,12
250 DATA1,14,"  EXAMPLE BAR CHART "
260 DATA20,2,0
270 DATA22,7,"1 2 3 4 5 6 7 8 9 1 1 1 1 1 1 "
280 DATA23,7,"                    0 1 2 3 4 5"
290 FORQ=1TO4
300 READA:POKE87,A:READA:POKE88,A:SYS(29216)
310 READA$:PRINT""A$
320 NEXTQ
330 REM
340 REM *DRAW BORDER AROUND DISPLAY
350 REM
360 POKE89,5:POKE90,2:POKE91,32:POKE92,20
365 POKE786,2:SYS(29024)
370 REM
380 REM *DEFINE SCALE FOR PLOTTING
390 REM
400 B=0
410 FORQ=1TO30
```

```
420 A=P%(Q)
430 IFA>BTHENB=A
440 NEXTQ
450 REM
460 REM *PRINT MAXIMUM DISPLAYED VALUE
470 REM
480 PRINT"▧▨▨▨▨▨◣"B
490 A=B/135
500 FORQ=1TO30
510 REM
520 REM *DISPLAY BARCHART #1 (BAR)
530 REM
540 AS=INT(P%(Q)/A)
545 C=INT(RND(1)*8)+1
546 IFC+240=PEEK(53281)ORC=CCTHEN545
550 POKE90,20:POKE89,Q+6:POKE91,AS:POKE786,C
560 SYS(29408)
565 CC=C
570 REM
580 REM *DISPLAY BARCHART #2 (LINE)
590 REM
600 Q=Q+1
610 AS=INT(P%(Q)/A)
615 C=INT(RND(1)*8)+1
616 IFC+240=PEEK(53281)ORC=CCTHEN615
620 POKE90,20:POKE89,Q+6:POKE91,AS:POKE786,C
630 SYS(29760)
635 CC=C
640 NEXTQ
650 GETA$:IFA$=""THEN 650
655 REM **WAIT TILL KEY PRESSED TO END**
READY.
```

138

SCROLLING ROUTINES

There are four machine code routines in this section. These are to scroll an area of the screen in the four (right, left, up and down) directions. The routines do a complete scroll of the required area and also scroll the colour RAM.
Again some of the previous routines are used for the displays and so are included in the loader.

SCREEN SCROLLING RIGHT

The first scrolling routine is called with SYS(29952); it scrolls a specified block of characters, one column to the right. The left most column is filled with spaces and the far right column disappears off the screen. The top left co-ordinates of the scrolled block are determined by two variables: column and line number. The width and height of the block are set by variables, with a minimum of 0 and a maximum equivalent to the dimensions of the screen. The routine requires the following variable locations:

   89 column number of top left of scrolled block.
   90 line number of top left of scrolled block.
   91 height of scrolled block width of scrolled block.
   92 width of scrolled block.
   The colour is retained throughout.

SCROLL LEFT

The second scrolling routine scrolls the contents of a set block of screen characters one column to the left, the right most column is filled with spaces and the left most column disappears. The top left co-ordinates of the scrolled block are determined by two variables: column and line number. The width and height of the block are set by variables, within a minimum of 0 and a maximum equivalent to the dimensions of the screen. The routine requires the following variable locations:

   89 column number of top left of scrolled block.
   90 line number of top left of scrolled block.
   91 height of scrolled block.
   92 width of scrolled block.
   The colour is retained throughout this routine.
   This routine is called with SYS (30976).

SCROLL UP

This routine scrolls a specified block of screen characters up one line, the bottom line is filled with spaces and the top line disappears. The top left co-ordinates of the scrolled block are determined by two variables: column number and line number. The width and height of the block are set by variables, with a minimum of 0 and a maximum equivalent to the dimensions of the screen. This routine requires the following variable locations:

89 column number of top left and scrolled block.
90 line number of top left of scrolled block.
91 width of scrolled block.
92 height of scrolled block.
The colour is retained throughout this routine.
This routine is called with SYS (30128).

SCROLL DOWN

This routine scrolls a specified block of screen characters down one line, the top line is filled with spaces and the bottom line disappears. The top left co-ordinates of the scrolled block are determined by two variables: column number and line number. The width and height of the block are set by variables, with a minimum of 0 and a maximum equivalent to the dimensions of the screen. This routine requires the following variable locations:

89 column number of top left and scrolled block.
90 line number of top left of scrolled block.
91 width of scrolled block.
92 height of scrolled block.
The colour is retained throughout this routine.
This routine is called with SYS (30240).

```
10 REM
20 REM ******************************
30 REM *ROUTINE TO CALCULATE SCREEN *
40 REM *ADDRESS FROM VALUES FOR     *
50 REM *    COLUMN      IN   89      *
60 REM *    LINE        IN   90      *
70 REM *ADDRESS RETURNED IN 91 AND 92*
80 REM *THIS ROUTINE IS ONLY CALLED  *
90 REM *BY OTHER ROUTINES IN THE     *
100 REM*PACKAGE.DO NOT USE BY ITSELF.*
110 REM*ERROR FLAG IN LOCATION 785   *
120 REM******************************
130 I=28672:T=0
140 READ A
150 IF A=-1 THEN 190
160 T=T+A:POKEI,A
170 I=I+1
180 GOTO 140
190 PRINT"{CLR}ROUTINE 1:"
200 IF T=11973 THEN PRINT"{DOWN}ENTERED O.K":GOTO 1000
210 PRINT"{DOWN}ENTERED INCORRECTLY"
220 END
230 DATA169,0,141,17,3,165,89
240 DATA48,66,201,40,176,74,165
250 DATA90,48,82,201,25,176,90
260 DATA169,0,133,87,133,88,133
270 DATA94,133,95,165,90,240,15
280 DATA170,24,165,87,105,40,133
290 DATA87,144,2,230,88,202,208
300 DATA242,24,165,87,101,89,133
310 DATA87,165,88,105,4,133,88
320 DATA24,165,87,133,94,165,88
330 DATA105,212,133,95,96,169,1
340 DATA141,17,3,169,0,133,89
350 DATA76,7,112,169,2,141,17
360 DATA3,169,39,133,89,76,7
370 DATA112,169,3,141,17,3,169
380 DATA0,133,90,76,7,112,169
390 DATA4,141,17,3,169,24,133
400 DATA90,76,7,112,-1
1000 REM
1010 REM******************************
1020 REM*DRAW A BORDER OF ANY SIZE,ANY*
1030 REM*LOCATION, AND ANY COLOUR. TOP*
1040 REM*LEFT COORDINATES ARE:        *
1050 REM*    COLUMN      IN   89       *
1060 REM*    LINE        IN   90       *
1070 REM*    WIDTH       IN   91       *
1080 REM*    HEIGHT      IN   92       *
1090 REM* COLOUR IS STORED IN 786.     *
1100 REM*ROUTINE CALLED BY SYS(29024) *
```

```
1110 REM**********************************
1120 I=29024:T=0
1130 READ A
1140 IF A=-1 THEN1180
1150 POKE I,A:T=T+A
1160 I=I+1
1170 GOTO1130
1180 PRINT"❏ROUTINE 2:"
1190 IF T=20557 THEN PRINT"❏❏ENTERED O.K.":GOTO2000
1200 PRINT"❏❏ENTERED INCORRECTLY"
1210 END
1220 DATA72,152,72,138,72,32,0
1230 DATA112,165,87,133,89,165,88
1240 DATA133,90,169,100,141,16,3
1250 DATA32,212,113,24,165,87,105
1260 DATA41,133,87,133,89,165,88
1270 DATA105,0,133,88,133,90,169
1280 DATA101,141,16,3,32,239,113
1290 DATA24,165,89,101,91,133,87
1300 DATA198,87,165,90,105,0,133
1310 DATA88,169,103,141,16,3,32
1320 DATA239,113,165,89,133,87,165
1330 DATA90,133,88,166,92,198,87
1340 DATA202,202,24,165,87,105,40
1350 DATA133,87,165,88,105,0,133
1360 DATA88,202,208,240,169,99,141
1370 DATA16,3,32,212,113,104,170
1380 DATA104,168,104,96,164,91,165
1390 DATA87,133,94,165,88,24,105
1400 DATA212,133,95,173,16,3,145
1410 DATA87,173,18,3,145,94,136
1420 DATA208,243,96,160,0,166,92
1430 DATA202,202,173,16,3,145,87
1440 DATA24,165,87,133,94,165,88
1450 DATA105,212,133,95,173,18,3
1460 DATA145,94,24,165,87,105,40
1470 DATA133,87,165,88,105,0,133
1480 DATA88,202,208,219,96,-1
2000 REM
2010 REM**********************************
2020 REM*ROUTINE TO PLACE THE CURSOR   *
2030 REM*AT A LOCATION ON THE SCREEN   *
2040 REM*WHOSE COORDINATES ARE STORED  *
2050 REM*IN  87=LINE AND 88=COLUMN.    *
2060 REM*ROUTINE  CALLED BY SYS(29216)*
2070 REM**********************************
2080 I=29216:T=0
2090 READ A
2100 IF A=-1 THEN2140
2110 POKE I,A:T=T+A
2120 I=I+1
```

```
2130 GOTO2090
2140 PRINT"ROUTINE 3:"
2150 IF T=5163 THEN PRINT"ENTERED O.K.":GOTO3000
2160 PRINT"ENTERED INCORRECTLY"
2170 END
2180 DATA72,152,72,138,72,169,19
2190 DATA32,22,231,165,87,240,9
2200 DATA169,17,32,22,231,198,87
2210 DATA208,247,165,88,240,9,169
2220 DATA29,32,22,231,198,88,208
2230 DATA247,104,170,104,168,104,96,-1
3000 REM
3010 REM*****************************
3020 REM*SCROLL CONTENTS OF SCREEN   *
3030 REM*BLOCK RIGHT ONE COLUMN. TOP *
3040 REM*LEFT COORDINATES OF BLOCK IN *
3050 REM*89 (COL) AND 90 (LINE). BLOCK*
3060 REM*HEIGHT IN 91 AND WIDTH IN 92.*
3070 REM*COLOUR IS RETAINED THROUGHOUT*
3080 REM*ROUTINE CALLED BY SYS(29952) *
3090 REM*****************************
3100 I=29952:T=0
3110 READ A
3120 IF A=-1 THEN3160
3130 POKE I,A:T=T+A
3140 I=I+1
3150 GOTO3110
3160 PRINT"ROUTINE 4:"
3170 IF T=10955 THEN PRINT"ENTERED O.K.":GOTO4000
3180 PRINT"ENTERED INCORRECTLY"
3190 END
3200 DATA72,152,72,138,72,32,0
3210 DATA112,56,165,87,233,1,133
3220 DATA89,165,88,233,0,133,90
3230 DATA166,91,164,92,136,177,89
3240 DATA145,87,165,90,24,105,212
3250 DATA133,97,165,88,24,105,212
3260 DATA133,95,165,89,133,96,165
3270 DATA87,133,94,177,96,145,94
3280 DATA136,208,223,169,32,145,87
3290 DATA24,165,87,105,40,133,87
3300 DATA144,2,230,88,24,165,89
3310 DATA105,40,133,89,144,2,230
3320 DATA90,202,208,191,104,170,104
3330 DATA168,104,96,-1
4000 REM
4010 REM*****************************
4020 REM*SCROLL CONTENTS OF SCREEN   *
4030 REM*BLOCK LEFT ONE COLUMN. TOP   *
4040 REM*LEFT COORDINATES OF BLOCK IN *
4050 REM*89 (COL) AND 90 (LINE). BLOCK*
```

```
4060 REM*HEIGHT IN 91 AND WIDTH IN 92.*
4070 REM*COLOUR IS RETAINED THROUGHOUT*
4080 REM*ROUTINE CALLED BY SYS(30976) *
4090 REM*****************************
4100 I=30976:T=0
4110 READ A
4120 IF A=-1 THEN4160
4130 POKE I,A:T=T+A
4140 I=I+1
4150 GOTO4110
4160 PRINT"⊃ROUTINE 5:"
4170 IF T=10720 THEN PRINT"▓▓ENTERED O.K.":GOTO5000
4180 PRINT"▓▓ENTERED INCORRECTLY"
4190 END
4200 DATA72,152,72,138,72,32,0
4210 DATA112,24,165,87,105,1,133
4220 DATA89,165,88,105,0,133,90
4230 DATA166,91,160,0,177,89,145
4240 DATA87,165,90,24,105,212,133
4250 DATA97,165,88,24,105,212,133
4260 DATA95,165,89,133,96,165,87
4270 DATA133,94,177,96,145,94,200
4280 DATA196,92,144,221,169,32,145
4290 DATA87,24,165,87,105,40,133
4300 DATA87,144,2,230,88,24,165
4310 DATA89,105,40,133,89,144,2
4320 DATA230,90,202,208,190,104,170
4330 DATA104,168,104,96,0,-1
5000 REM
5010 REM*****************************
5020 REM*SCROLL CONTENTS OF SCREEN   *
5030 REM*BLOCK UP ONE LINE. TOP LEFT *
5040 REM*COORDINATES OF BLOCK IN 89  *
5050 REM*(COL) AND 90 (LINE). BLOCK  *
5060 REM*HEIGHT IN 91 AND WIDTH IN 92.*
5070 REM*COLOUR IS RETAINED THROUGHOUT*
5080 REM*ROUTINE CALLED BY SYS(30128) *
5090 REM*****************************
5100 I=30128:T=0
5110 READ A
5120 IF A=-1 THEN5160
5130 POKE I,A:T=T+A
5140 I=I+1
5150 GOTO5110
5160 PRINT"⊃ROUTINE 6:"
5170 IF T=11564 THEN PRINT"▓▓ENTERED O.K.":GOTO6000
5180 PRINT"▓▓ENTERED INCORRECTLY"
5190 END
5200 DATA72,152,72,138,72,32,0
5210 DATA112,24,165,87,105,40,133
5220 DATA89,165,88,105,0,133,90
```

```
5230 DATA166.92.164.91.136.177.89
5240 DATA145.87.165.90.24.105.212
5250 DATA133.97.165.88.24.105.212
5260 DATA133,95,165,89,133,96,165
5270 DATA87,133,94,177,96,145,94
5280 DATA136,16,223,202,240,25,24
5290 DATA165,87,105,40,133,87,144
5300 DATA2,230,88,24,165,89,105
5310 DATA40,133,89,144,2,230,90
5320 DATA76,199,117,164,91,136,169
5330 DATA32,145,87,136,16,249,104
5340 DATA170,104,168,104,96,-1
6000 REM
6010 REM*****************************
6020 REM*SCROLL CONTENTS OF SCREEN  *
6030 REM*BLOCK DOWN ONE LINE. TOP LEFT*
6040 REM*COORDINATES OF BLOCK IN 89  *
6050 REM*(COL) 90 (LINE).BLOCK HEIGHT *
6060 REM*IN 91 AND WIDTH IN 92.     *
6070 REM*COLOUR IS RETAINED THROUGHOUT*
6080 REM*ROUTINE CALLED BY SYS(30240) *
6090 REM*****************************
6100 I=30240:T=0
6110 READ A
6120 IF A=-1 THEN6160
6130 POKE I,A:T=T+A
6140 I=I+1
6150 GOTO6110
6160 PRINT":ROUTINE 7:"
6170 IF T=14455 THEN PRINT"ENTERED O.K.":GOTO7000
6180 PRINT"ENTERED INCORRECTLY"
6190 END
6200 DATA72,152,72,138,72,32,0
6210 DATA112,166,92,202,24,165,87
6220 DATA105,40,133,87,144,2,230
6230 DATA88,202,208,242,56,165,87
6240 DATA233,40,133,89,165,88,233
6250 DATA0,133,90,166,92,164,91
6260 DATA136,177,89,145,87,165,90
6270 DATA24,105,212,133,97,165,88
6280 DATA24,105,212,133,95,165,89
6290 DATA133,96,165,87,133,94,177
6300 DATA96,145,94,136,16,223,202
6310 DATA240,25,56,165,87,233,40
6320 DATA133,87,176,2,198,88,56
6330 DATA165,89,233,40,133,89,176
6340 DATA2,198,90,76,72,118,164
6350 DATA91,136,169,32,145,87,136
6360 DATA208,249,104,170,104,168,104
6370 DATA96,-1
7000 REM
```

```
7010 REM*********************************
7020 REM*DRAW VERTICAL BAR - BOTTOM TO*
7030 REM*TOP OF SPACES TERMINATED IN A*
7040 REM*THIN LINE,RESOLUTION 135 X 40*
7050 REM*COLUMN # IN 89,             *
7060 REM*LINE   # IN 90.             *
7070 REM*LENGTH   IN 91.             *
7080 REM*COLOUR   IN 786.            *
7090 REM*ROUTINE CALLED BY SYS(29760) *
7100 REM*********************************
7110 I=29760:T=0
7120 READ A
7130 IF A=-1 THEN7170
7140 POKE I,A:T=T+A
7150 I=I+1
7160 GOTO7120
7170 PRINT"⌑ROUTINE 8:"
7180 IF T=8307 THEN PRINT"▨▨ENTERED O.K.":GOTO7330
7190 PRINT"▨▨ENTERED INCORRECTLY"
7200 END
7210 DATA72,152,72,138,72,32,0
7220 DATA112,160,0,165,91,201,8
7230 DATA48,37,233,8,133,91,169
7240 DATA32,145,87,173,18,3,145
7250 DATA94,24,165,87,233,39,133
7260 DATA87,133,94,165,88,233,0
7270 DATA133,88,24,105,212,133,95
7280 DATA165,88,208,213,170,189,134
7290 DATA116,145,87,173,18,3,145
7300 DATA94,104,170,104,168,104,96
7310 DATA100,82,70,64,67,68,69
7320 DATA99,32,0,-1
7330 POKE51,0:POKE52,112
7340 POKE55,0:POKE56,112:CLR:NEW
READY.
```

146

BLOCK SCROLLING

The loader on pp. 141-6 should be loaded and run once before using this program.

DESCRIPTION

The program uses a block scroll routine to scroll up or down a long list of items. To scroll up use the F1 key, and to scroll down use the F3 key. The program uses the following Machine Code routines:

DRAW BORDER – SYS (29024)
SCROLL BLOCK UP OR DOWN ONE LINE – SYS (30128)
CURSOR CONTROL – SYS (29216)

PROGRAM STRUCTURE

The following lines are of particular interest:

120-135	: Limit the top of memory and clear the pointers.
210-320	: The data table to be displayed.
330-360	: Reads the data into the array L$.
400-470	: Displays the test using the cursor control routine.
505-520	: Draws a border around the display.
590-640	: Checks for legal key press and if without bounds scrolls up or down.
680-710	: Scrolls the list up one entry.
750-780	: Scrolls the list down one entry.

```
10 REM *******************************
20 REM *THIS PROGRAM USES THE UP AND  *
25 REM *DOWN BLOCK SCROLL             *
30 REM *ROUTINES TO DISPLAY THE       *
35 REM *CONTENTS OF A LONG LIST       *
40 REM *DATA ITEMS. USE KEY 8 TO      *
45 REM *SCROLL UP THE LIST            *
50 REM *AND KEY 2 TO SCROLL DOWN THE  *
55 REM *LIST.                         *
60 REM *THE MACHINE CODE ROUTINES USED*
65 REM *BY THIS PROGRAM ARE:          *
70 REM *   SYS(29024) - DRAW A BORDER *
80 REM *   SYS(30128) - SCROLL BLOCK  *
85 REM *    UP ONE LINE               *
90 REM *    DOWN ONE LINE             *
100 REM *  SYS(29216) - CURSOR CONTROL*
110 REM *******************************
120 POKE55,0:POKE56,112
130 POKE51,0:POKE52,112:CLR
135 REM LOWER TOP OF MEMORY
140 DIML$(40)
150 REM
160 REM *TABLE OF DATA TO BE DISPLAYED.
165 REM *  NOTE THE USE OF
170 REM *NULL ENTRIES AT BEGINING AND
175 REM *  END OF THE TABLE
180 REM *THIS GIVES A BLANK SCREEN
185 REM *  TRAILER AT BOTH ENDS
190 REM *OF THE DISPLAY. ESSENTIAL FOR
195 REM *     NEATNESS.
200 REM
210 DATA" "," "," "," "," "," "," "," "
220 DATA "ITEM 1 ......................"
225 DATA "ITEM 2 ......................"
230 DATA "ITEM 3 ......................"
235 DATA "ITEM 4 ......................"
240 DATA "ITEM 5 ......................"
245 DATA "ITEM 6 ......................"
250 DATA "ITEM 7 ......................"
255 DATA "ITEM 8 ......................"
260 DATA "ITEM 9 ......................"
265 DATA "ITEM 10 ......................"
270 DATA "ITEM 11 ......................"
275 DATA "ITEM 12 ......................"
280 DATA "ITEM 13 ......................"
285 DATA "ITEM 14 ......................"
290 DATA "ITEM 15 ......................"
295 DATA "ITEM 16 ......................"
300 DATA "ITEM 17 ......................"
305 DATA "ITEM 18 ......................"
310 DATA "ITEM 19 ......................"
```

```
315 DATA "ITEM 20 ....................."
320 DATA" "," "," "," "," "," "," "," "
330 FORQ=1TO36
340 READA$
350 L$(Q)=A$
360 NEXTQ
370 REM
380 REM *DISPLAY TEXT USING CURSOR
385 REM *        CONTROL
390 REM
400 PRINT"⌧"
405 POKE53281,12:POKE53280,12
410 DATA3,6,"⧉EXAMPLE OF SCROLLED LONG LIST█"
420 DATA7,9,"⧉USE KEY F1 TO SCROLL UP   █"
430 DATA8,9,"⧉USE KEY F3 TO SCROLL DOWN█"
440 FORQ=1TO3
450 READA:POKE87,A:READA:POKE88,A:SYS(29216)
460 READA$:PRINT"⧉"A$
470 NEXTQ
480 REM
490 REM *DRAW BORDER AROUND SCROLLED
495 REM *  PORTION OF SCREEN
500 REM
505 C=0
510 POKE89,0:POKE90,10:POKE91,39:POKE92,10
515 POKE786,C:SYS(29024)
520 GOTO680
530 REM
540 REM *LOOK FOR KEY INPUT IN
545 REM *LOCATION 197
550 REM *IF 197=4 THEN KEY F1 PRESSED
555 REM *   AND SCROLL UP
560 REM *IF 197=5 THEN KEY F3 PRESSED
565 REM *   AND SCROLL DOWN
570 REM *ALSO CHECK POINTER TO TABLE IS
575 REM *  WITHIN BOUNDS.
580 REM
590 IFQ<1THENQ=1
600 IFQ>29THENQ=29
610 X=PEEK(197)
620 IFX=4 THEN Q=Q+1:GOTO680
630 IFX=5 THEN Q=Q-1:GOTO750
640 GOTO590
650 REM
660 REM *SCROLL LIST UP ONE ENTRY
670 REM
680 POKE89,2:POKE90,12:POKE91,37:POKE92,6
685 SYS(30128)
690 POKE87,17:POKE88,2:SYS(29216)
700 PRINT"⧉"L$(Q+5)
710 GOTO590
```

```
720 REM
730 REM *SCROLL LIST DOWN ONE ENTRY
740 REM
750 POKE89,2:POKE90,12:POKE91,37:POKE92,6
755 SYS(30240)
760 POKE87,12:POKE88,2:SYS(29216)
770 PRINT"█"L$(Q)
780 GOTO590
READY.
```

LONG WAVEFORM GRAPH

The loader on pp. 141-6 should be loaded and run once before using this program.

DESCRIPTION

This program displays the graph of a function and scrolls the function across the screen from right to left. The Machine Code routines used in this program are as follows:

DRAW BORDER – SYS (29024)
CURSOR CONTROL – SYS (29216)
SCROLL SCREEN BLOCK LEFT – SYS (30976)
DRAW VERTICAL FINE RES LINE – SYS (29760)

PROGRAM STRUCTURE

The lines of interest in the program are as follows:

19-20	:	Limit top of memory and clear pointers.
40-41	:	Draw border around the scrolled screen area.
50-80	:	Displays the text using the cursor control routine.
90-140	:	Displays the graph of the function and scrolls from left to right.

```
1 REM *******************************
2 REM *THIS IS A VARIATION ON THE PRG *
3 REM *ON PAGE 132, THE GRAPH OF THE  *
4 REM *FUNCTION IS SCROLLED ACROSS    *
5 REM *THE SCREEN FROM RIGHT          *
6 REM *TO LEFT CONTINUOUSLY AS EACH   *
7 REM *PLOT POINT IS CALCULATED.      *
8 REM *IDEAL FOR DISPLAYING LONG      *
9 REM *GRAPHICAL DISPLAYS.            *
10 REM*USES THE FOLLOWING MACHINE     *
11 REM*CODE ROUTINES:                 *
12 REM*  SYS(29216) - CURSOR CONTROL  *
13 REM*  SYS(29024) - DRAW BORDER     *
14 REM*  SYS(30976) - SCROLL SCREEN   *
15 REM*  BLOCK LEFT                   *
16 REM*   SYS(29760) - DRAW VERTICAL  *
17 REM*  FINE RESOLUTION LINE         *
18 REM*******************************
19 POKE55,0:POKE56,112
20 POKE51,0:POKE52,112:CLR
25 REM LOWER TOP OF MEMORY POINTERS
30 PRINT"⌑"
31 POKE53281,12:POKE53280,12
34 REM
35 REM *DRAW BORDER AROUND SCROLLED
36 REM *   SCREEN AREA
37 REM
40 POKE89,0:POKE90,7:POKE91,39:POKE92,15
41 POKE786,0:SYS(29024)
44 REM
45 REM *DISPLAY TEXT USING CURSOR
46 REM *    CONTROL
47 REM
50 POKE87,3:POKE88,4:SYS(29216)
60 PRINT"▓▒EXAMPLE DISPLAY OF LONG WAVEFORM█"
70 POKE87,4:POKE88,10:SYS(29216)
80 PRINT"▓▒USING LEFT SCROLLING█"
84 REM
85 REM *DISPLAY GRAPH OF FUNCTION
86 REM *SCROLLED FROM RIGHT TO LEFT
87 REM
90 FORQ=1TO100STEP.15
100 A=INT(SIN(Q)*40)+45
110 POKE89,2:POKE90,8:POKE91,12:POKE92,36
115 SYS(30976)
120 POKE89,38:POKE90,19:POKE91,A:POKE786,6
125 SYS(29760)
130 FORX=1TO100:NEXTX
140 NEXTQ
READY.
```

152

LARGE SCREEN DISPLAY

LARGE SCREEN INITIALISATION

The area to be used as a large screen is first initialised, the variables to be used are initialised and the memory area, to be used for text storage, is cleared. There are three variables that the initialisation sequence uses. They are as follows: the number of columns wide and lines deep of the large screen and the most significant byte of the starting address of the large screen memory. The top of the memory must be lowered to the start of the large screen memory, otherwise Basic will overwrite the text area. The starting address used will depend on the size of the large text area, and each page will require 1K of memory. In our example Basic program, this will allow 8 pages of text. This routine requires the following variable locations:

820 large screen width in columns.

821 large screen depth in lines.

823 MSB of start of text storage memory address.

This routine is called from a Basic program with SYS (31083)

The large screen section also uses some of the other Machine Code routines so all the code should be in the 64 before using.

LARGE SCREEN CLEAR

This section of code clears the entire large screen memory area, all locations in that area are set to decimal 32, the screen ASCII code for a space character. This routine does not require any variables.

To call this routine from a Basic program use SYS (31124).

LARGE SCREEN DISPLAY

Since the text area is actually larger than the size of the screen, this routine displays a section of the large screen. Specifying that the co-ordinates are column 0 and line 0, they will display the left 40 columns and the top 25 lines of the large screen text area. Similarly column 40 and line 0 will display the right hand 40 columns and the top 25 lines, or column 20 line 0 will display the central 40 columns and the top 25 lines. Using this procedure any part of the text area can be displayed, providing it is within the limits of the maximum column and line number. Line and column number should not be less than 0, nor should the column position be greater than the large screen width minus 40, or the line number greater than the large screen width minus 40, or the line number

greater than the large screen depth minus 25. These co-ordinates designate the top left corner of the screen, the use of greater values will result in a faulty display. This routine requires the following variable locations:

824 large screen column number where top left of display starts.
825 large screen line number where top left of display starts.
This routine is called from a Basic program with SYS(31173).

LARGE SCREEN SAVE

This section of code saves the contents of the screen and stores it in the large screen memory area, in a position corresponding to the co-ordinates specified. The co-ordinates refer to the line and column number of the large screen and designate the position of the top left corner of the actual 64 screen. The constraints on co-ordinate size are the same as that defined for large screen display. Text displayed in the large screen memory can be saved and then recalled later, using a monitor to save it. This routine requires the following variable locations:

824 large screen column number where top left of display starts.
825 large screen line number where top left of display starts.
This routine is called from a Basic program with SYS (31261).

DISPLAY PAGE

This routine requires the large screen memory to be initialised with the initialisation routine so that it is 40 columns wide and up to six pages or 150 lines long. Any of the six pages can be displayed by putting the required page number into location 91 and calling the routine. If fewer pages are required, then the initialisation routine should be given the correct number of lines for the required number of pages. This routine requires the following variable locations:

91 page number to be displayed.
To call this routine from a Basic program use SYS (31349).

SAVE LARGE SCREEN PAGE

This routine will save the current screen contents into the large screen memory area at a position determined by the page number. This routine requires the large screen memory to be initialised with the initialisation routine so that it is 40 columns wide and up to six pages or 150 lines long. Any of the six pages can be saved by putting the required page number into location 91 and calling the routine. If fewer pages are required, then the initialisation routine should be given the correct number of lines for the required

number of pages. This routine requires the following variable locations:

91 page number to be saved.

To call this routine from a Basic program use SYS(31381).

```
10 REM
20 REM ******************************
30 REM *ROUTINE TO CALCULATE SCREEN  *
40 REM *ADDRESS FROM VALUES FOR      *
50 REM *    COLUMN     IN   89       *
60 REM *    LINE       IN   90       *
70 REM *ADDRESS RETURNED IN 91 AND 92*
80 REM *THIS ROUTINE IS ONLY CALLED  *
90 REM *BY OTHER ROUTINES IN THE     *
100 REM*PACKAGE,DO NOT USE BY ITSELF.*
110 REM*ERROR FLAG IN LOCATION 785   *
120 REM******************************
130 I=28672:T=0
140 READ A
150 IF A=-1 THEN 190
160 T=T+A:POKEI,A
170 I=I+1
180 GOTO 140
190 PRINT"⊃ROUTINE 1:"
200 IF T=11973 THEN PRINT"▓▓ENTERED O.K":GOTO 1000
210 PRINT"▓▓ENTERED INCORRECTLY"
220 END
230 DATA169,0,141,17,3,165,89
240 DATA48,66,201,40,176,74,165
250 DATA90,48,82,201,25,176,90
260 DATA169,0,133,87,133,88,133
270 DATA94,133,95,165,90,240,15
280 DATA170,24,165,87,105,40,133
290 DATA87,144,2,230,88,202,208
300 DATA242,24,165,87,101,89,133
310 DATA87,165,88,105,4,133,88
320 DATA24,165,87,133,94,165,88
330 DATA105,212,133,95,96,169,1
340 DATA141,17,3,169,0,133,89
350 DATA76,7,112,169,2,141,17
360 DATA3,169,39,133,89,76,7
370 DATA112,169,3,141,17,3,169
380 DATA0,133,90,76,7,112,169
390 DATA4,141,17,3,169,24,133
400 DATA90,76,7,112,-1
1000 REM
1010 REM
1020 REM******************************
1030 REM*DRAW A HORIZONTAL BAR OF ANY *
1040 REM*64 CHARACTER.CHARACTER STORED*
1050 REM*IN 784, COLUMN START IN 89,  *
1060 REM*LINE START IN 90, LENGTH OF  *
1070 REM*LINE IN 91, COLOUR IN 786.   *
1080 REM*ROUTINE CALLED BY SYS(28800) *
1090 REM******************************
1100 I=28800:T=0
```

```
1110 READ A
1120 IF A=-1 THEN 1160
1130 POKE I,A:T=T+A
1140 I=I+1
1150 GOTO 1110
1160 PRINT"⊐ROUTINE 2:"
1170 IF T=3095 THEN PRINT"⬛⬛ENTERED O.K":GOTO 2000
1180 PRINT"⬛⬛ENTERED INCORRECTLY"
1190 END
1200 DATA72,152,72,138,72,32,0
1210 DATA112,164,91,173,16,3,145
1220 DATA87,173,18,3,145,94,136
1230 DATA208,243,104,170,104,168,104
1240 DATA96,-1
2000 REM
2010 REM****************************
2020 REM*INVERT A BLOCK OF THE SCREEN *
2030 REM*TOP LEFT COORDINATES IN 89   *
2040 REM*(COL) AND 90(LINE), HEIGHT IN*
2050 REM*91 AND WIDTH IN 92, COLOUR IN*
2060 REM*786.                         *
2070 REM*ROUTINE CALLED BY SYS(28960) *
2080 REM****************************
2090 I=28960:T=0
2100 READ A
2110 IF A=-1 THEN2150
2120 POKE I,A:T=T+A
2130 I=I+1
2140 GOTO2100
2150 PRINT"⊐ROUTINE 3:"
2160 IF T=5859 THEN PRINT"⬛⬛ENTERED O.K.":GOTO3000
2170 PRINT"⬛⬛ENTERED INCORRECTLY"
2180 END
2190 DATA72,152,72,138,72,32,0
2200 DATA112,166,92,164,91,136,177
2210 DATA87,73,128,145,87,173,18
2220 DATA3,145,94,136,16,242,24
2230 DATA165,87,105,40,133,87,133
2240 DATA94,144,4,230,88,230,95
2250 DATA202,208,221,104,170,104,168
2260 DATA104,96,-1
3000 REM
3010 REM****************************
3020 REM*DRAW A BORDER OF ANY SIZE,ANY*
3030 REM*LOCATION, AND ANY COLOUR. TOP*
3040 REM*LEFT COORDINATES ARE:        *
3050 REM*    COLUMN      IN    89      *
3060 REM*    LINE        IN    90      *
3070 REM*    WIDTH       IN    91      *
3080 REM*    HEIGHT      IN    92      *
3090 REM* COLOUR IS STORED IN 786.     *
```

```
3100 REM*ROUTINE CALLED BY SYS(29024) *
3110 REM*****************************
3120 I=29024:T=0
3130 READ A
3140 IF A=-1 THEN3180
3150 POKE I,A:T=T+A
3160 I=I+1
3170 GOTO3130
3180 PRINT"ROUTINE 4:"
3190 IF T=20557 THEN PRINT"ENTERED O.K.":GOTO4000
3200 PRINT"ENTERED INCORRECTLY"
3210 END
3220 DATA72,152,72,138,72,32,0
3230 DATA112,165,87,133,89,165,88
3240 DATA133,90,169,100,141,16,3
3250 DATA32,212,113,24,165,87,105
3260 DATA41,133,87,133,89,165,88
3270 DATA105,0,133,88,133,90,169
3280 DATA101,141,16,3,32,239,113
3290 DATA24,165,89,101,91,133,87
3300 DATA198,87,165,90,105,0,133
3310 DATA88,169,103,141,16,3,32
3320 DATA239,113,165,89,133,87,165
3330 DATA90,133,88,166,92,198,87
3340 DATA202,202,24,165,87,105,40
3350 DATA133,87,165,88,105,0,133
3360 DATA88,202,208,240,169,99,141
3370 DATA16,3,32,212,113,104,170
3380 DATA104,168,104,96,164,91,165
3390 DATA87,133,94,165,88,24,105
3400 DATA212,133,95,173,16,3,145
3410 DATA87,173,18,3,145,94,136
3420 DATA208,243,96,160,0,166,92
3430 DATA202,202,173,16,3,145,87
3440 DATA24,165,87,133,94,165,88
3450 DATA105,212,133,95,173,18,3
3460 DATA145,94,24,165,87,105,40
3470 DATA133,87,165,88,105,0,133
3480 DATA88,202,208,219,96,-1
4000 REM
4010 REM*****************************
4020 REM*ROUTINE TO PLACE THE CURSOR  *
4030 REM*AT A LOCATION ON THE SCREEN  *
4040 REM*WHOSE COORDINATES ARE STORED *
4050 REM*IN  87=LINE AND 88=COLUMN.   *
4060 REM*ROUTINE  CALLED BY SYS(29216)*
4070 REM*****************************
4080 I=29216:T=0
4090 READ A
4100 IF A=-1 THEN4140
4110 POKE I,A:T=T+A
```

158

```
4120 I=I+1
4130 GOTO4090
4140 PRINT"DROUTINE 5:"
4150 IF T=5163 THEN PRINT"WENTERED O.K.":GOTO5000
4160 PRINT"WENTERED INCORRECTLY"
4170 END
4180 DATA72,152,72,138,72,169,19
4190 DATA32,22,231,165,87,240,9
4200 DATA169,17,32,22,231,198,87
4210 DATA208,247,165,88,240,9,169
4220 DATA29,32,22,231,198,88,208
4230 DATA247,104,170,104,168,104,96,-1
5000 REM
5010 REM
5020 REM**************************************
5030 REM* THIS ROUTINE ENABLES THE USE OF A    *
5040 REM* LARGE SCREEN IN MEMORY USING THE     *
5050 REM* NORMAL VIDEO SCREEN AS A WINDOW ON   *
5060 REM* THE LARGE SCREEN. THE DIMENSIONS OF  *
5070 REM* THE LARGE SCREEN ARE 80 COLUMNS BY   *
5080 REM* 100 LINES AND THE WINDOW WILL BE     *
5090 REM* MOVED OVER THE LARGE SCREEN AS THE   *
5100 REM* DATA IS ENTERED.                     *
5110 REM* THE CALLS FOR THE ROUTNIES           *
5120 REM* INCORPORATED ARE AS FOLLOWS          *
5130 REM*  1) SET UP LARGE SCREEN POINTERS     *
5140 REM*      LINES 120-150 OF EXAMPLE        *
5150 REM*  2) CALL LARGE SCREEN DISPLAY ROUTINE*
5160 REM*      LINES 640-650 OF EXAMPLE        *
5170 REM*  3) CLEAR LARGE SCREEN               *
5180 REM*      LINE 1340 OF EXAMPLE            *
5190 REM*  4) LARGE SCREEN SAVE ROUTINE        *
5200 REM*      LINE 1580 OF EXAMPLE            *
5210 REM*                                      *
5220 REM**************************************
5230 I=30957:E=0:PRINT"DROUTINE 6:"
5240 T=0
5250 READ A
5260 IF A=-1 THEN5380
5270 IF A>255 THEN5310
5280 T=T+A:POKE I,A
5290 I=I+1
5300 GOTO5250
5310 READ A$
5320 IF T<>A THEN5350
5330 PRINTA$"O.KD"
5340 GOTO5240
5350 E=E+1
5360 PRINTA$:T:A
5370 GOTO5240
5380 IF E<>0 THEN END
```

```
5390 POKE51,0:POKE52,112
5400 POKE55,0:POKE56,112:CLR:NEW
5410 DATA169,48,141,58,3,165,89
5420 DATA48,55,205,52,3,176,65
5430 DATA165,90,48,77,205,53,3
5440 DATA176,87,169,0,133,87,133
5450 DATA88,165,90,240,16,170,24
5460 DATA165,87,109,54,3,133,87
5470 DATA144,2,230,88,202,208,241
5480 DATA24,165,87,101,89,133,87
5490 DATA165,88,109,55,3,133,88
5500 DATA96,169,57,141,58,3,169
5510 DATA 7269,"LINES 3440 - 3530 "
5520 DATA0,133,89,141,56,3,76
5530 DATA242,120,169,57,141,58,3
5540 DATA173,52,3,133,89,141,56
5550 DATA3,76,242,120,169,57,141
5560 DATA58,3,169,0,133,90,141
5570 DATA57,3,76,242,120,169,57
5580 DATA141,58,3,173,53,3,133
5590 DATA90,141,57,3,76,242,120
5600 DATA72,152,72,138,72,165,1
5610 DATA41,254,133,1,169,0,141
5620 DATA 6765,"LINES 3550 - 3640 "
5630 DATA56,3,141,57,3,173,52
5640 DATA3,141,54,3,233,40,141
5650 DATA52,3,173,53,3,233,25
5660 DATA141,53,3,76,159,121,72
5670 DATA152,72,138,72,165,1,41
5680 DATA254,133,1,169,0,133,87
5690 DATA173,55,3,133,88,169,32
5700 DATA160,0,145,87,136,208,251
5710 DATA230,88,165,88,201,112,208
5720 DATA239,165,1,9,1,133,1
5730 DATA 6966,"LINES 3660 - 3750 "
5740 DATA104,170,104,168,104,96,72
5750 DATA152,72,138,72,165,1,41
5760 DATA254,133,1,173,56,3,133
5770 DATA89,173,57,3,133,90,32
5780 DATA237,120,169,0,133,89,169
5790 DATA4,133,90,162,23,160,39
5800 DATA32,181,122,177,87,145,89
5810 DATA177,94,145,96,136,16,245
5820 DATA24,165,87,109,54,3,133
5830 DATA87,144,2,230,88,24,165
5840 DATA 7374,"LINES 3770 - 3860 "
5850 DATA89,105,40,133,89,144,2
5860 DATA230,90,202,208,214,165,1
5870 DATA9,1,133,1,104,170,104
5880 DATA168,104,96,72,152,72,138
5890 DATA72,165,1,41,254,133,1
```

```
5900 DATA173.56.3,133,89,173,57
5910 DATA3,133,90.32,237,120.169
5920 DATA0,133,89,169.4,133.90
5930 DATA162.23.160,39,32,181,122
5940 DATA177.89.145.87,177,96,145
5950 DATA 7424,"LINES 3880 - 3970 "
5960 DATA94,136,16,245.24.165.87
5970 DATA109.54.3,133,87,144,2
5980 DATA230.88.24.165.89.105.40
5990 DATA133,89.144.2.230.90.202
6000 DATA208,214,165.1.9.1.133
6010 DATA1.104.170,104,168,104,96
6020 DATA72.152.72.138.72,165,1
6030 DATA41,254,133,1.169.0,133
6040 DATA89,133,90,24.166,91.165
6050 DATA90.105.24,133,90,202,208
6060 DATA 7421,"LINES 3990 - 4080 "
6070 DATA247,76.218.121,72.152.72
6080 DATA138,72,165.1.41.254.133
6090 DATA1,169,0.133.89,133,90
6100 DATA24.166.91,165,90,105.24
6110 DATA133,90,202,208,247,76.50
6120 DATA122,165.87,133.94.165.89
6130 DATA133,96,165,88.24.105.80
6140 DATA133,95,165.90,24,105.212
6150 DATA133.97.96
6160 DATA 6744,"LINES 4100 - 4180 ",-1
READY.
```

MULTIPLE SCREEN DISPLAY

The loader on pp. 156-61 should be loaded and run once before using this program.

DESCRIPTION

This program is a demo of multiple page display and save. The program is the Machine Code routines to set up and manipulate the screen. Up to six pages can be displayed and stored.

PROGRAM STRUCTURE

The following lines are of interest:

60-70 : Limit top of memory and clear the pointers.
105-140 : Set screen and border colours and initialise pointers for the large screen area.
250-450 : Creates the six pages and awaits user input and command to move to next page.
520-610 : Displays any of the six pages created.

```
10 REM ********************************
20 REM *DEMONSTRATION OF MULTIPLE     *
25 REM *SCREEN PAGE SAVE AND DISPLAY  *
30 REM *USING LARGE SCREEN ROUTINES.  *
35 REM *WILL SAVE UP TO SIX SEPERATE  *
40 REM *SCREEN PAGES.                 *
50 REM ********************************
60 POKE55,0:POKE56,88
70 POKE51.0:POKE52,88:CLR
80 REM
90 REM INITIALISE VARIABLE POINTERS FOR
95 REM      LARGE SCREEN
100 REM
105 POKE53281,12:POKE53280.12
110 POKE820.40
120 POKE821.150
130 POKE823,89
140 SYS(31083)
150 REM
160 REM
170 REM
180 REM CREATE SIX DEMONSTRATION SCREEN
185 REM  PAGES AND STORE FOR LATER
190 REM RECALL. ENTER ANY TEXT OF YOUR
195 REM   OWN AND THEN PRESS 'HOME'
200 REM TO SAVE THAT PAGE IN MEMORY.
210 REM
220 PRINT"⬛"
230 DATA"DEMO PAGE ONE","DEMO PAGE TWO"
235 DATA"DEMO PAGE THREE"
240 DATA"DEMO PAGE FOUR","DEMO PAGE FIVE"
245 DATA"DEMO PAGE SIX"
250 POKE87.5:POKE88.5:SYS(29216)
260 PRINT"⬛CREATE DEMONSTRATION PAGES AND"
270 POKE87.6:POKE88.12:SYS(29216)
280 PRINT"⬛STORE IN MEMORY"
290 POKE87.10:POKE88.4:SYS(29216)
300 PRINT"⬛PRESS 'HOME' TO CREATE NEXT PAGE⬛":
310 GETA$:IFA$=""THEN310
320 IFA$<>"⬛"THEN310
330 PRINT"⬛"
340 FORQ=1TO6
350 A=Q+48
360 POKE89,0:POKE90.0:POKE91.39:POKE92,23
365 POKE786.6:SYS(29024)
370 POKE87.10:POKE88.13:SYS(29216)
380 READA$:PRINTA$
385 POKE87,2:POKE88.2:SYS(29216)
390 GETA$:IFA$=""THEN390
400 IFA$="⬛"THEN430
410 IF A$<>CHR$(13)THENPRINTA$::GOTO420
```

```
415 PRINTA$;"██";
420 GOTO390
430 POKE91,Q:SYS(31381)
440 PRINT"⊐"
450 NEXTQ
460 REM
470 REM
480 REM
490 REM
500 REM DISPLAY ANY OF THE PREVIOUSLY
505 REM    CREATED SIX PAGES.
510 REM
520 POKE87.10:POKE88.5:SYS(29216)
530 PRINT"█▆DISPLAY DEMONSTRATION PAGES█"
540 POKE 89.0:POKE90.23:POKE91.40:POKE784.32
545 POKE786.6:SYS(28800)
550 POKE 89.0:POKE90.24:POKE91.40:POKE784.32
555 POKE786.6:SYS(28800)
560 POKE87.23:POKE88.0:SYS(29216)
570 PRINT"█▜PAGE # ";
580 INPUTA
590 IFA<1ORA>6THEN540
600 POKE91.A:SYS(31349)
610 GOTO540
READY.
```

```
THIS IS A SAMPLE SCREEN PRODUCED WITH

WITH THIS PROGRAM AND STORED IN MEMORY

           FOR LATER RECALL.


   UP TO SIX SCREENS MAY BE KEPT IN

           MEMORY AT ONE TIME.
```

LARGE SCREEN

The loader on pp. 156-61 should be loaded and run once before using this program.

DESCRIPTION

This program is a demo of the routines to control a large screen text area. It emulates some of the functions of a simple word processor, although the speed is restricted as the bulk of the program is written in Basic.

PROGRAM STRUCTURE

The lines of particular interest in the program are as follows:

65-80	:	Set the screen and border colours and limit the top of memory, then clear the pointers.
120-150	:	Initialises the pointers for the large screen area.
210-230	:	Puts the cursor at the top left and sets up the control characters.
270-320	:	Scans for cursor control inputs.
390-410	:	Scans for the F1 key. If pressed, highlights the C and allows access to the control options, which are:
		I – input data to the screen.
		0 – output to the printer.
		D – delete the entire contents of large screen area.
450-520	:	Displays the control line.
580-670	:	This is the cursor control subroutine. This is the cursor up routine.
710-800	:	This is the cursor left routine.
840-930	:	This is the cursor right routine.
970-1060	:	This is the cursor down routine.
1100-1160	:	This is the cursor home routine.
1190-1250	:	Controls the input for the Control keys, F1 exits this mode.
1290-1390	:	Executes the large screen erase.
1430-1510	:	Routine to put data onto the large screen.
1770-1910	:	Routine to output contents of the large screen to the printer.

```
10 REM ***************************
20 REM *DEMONSTRATION OF ROUTINES*
25 REM *TO CONTROL LARGE SCREEN  *
30 REM *TEXT AREA. THE PROGRAM   *
35 REM *EMULATES SOME OF THE     *
40 REM *FUNCTIONS OF A SIMPLE    *
45 REM *WORD PROCESSOR SINCE MOST*
50 REM *OF THE CODE IS WRITTEN IN*
55 REM *BASIC IT IS VERY SLOW    *
60 REM ***************************
65 POKE53281,12:POKE53280.12:Z=1
70 POKE55,0:POKE56,80
80 POKE51.0:POKE52.80:CLR
90 REM
100 REM INITIALISE POINTERS FOR LARGE SCREEN
110 REM
120 POKE820,82
130 POKE821,77
140 POKE823,80
150 SYS(31083)
160 LP=0:CP=0:CC=0:LC=0:C=0:P=0
170 REM
180 REM PUT CURSOR AT TOP LEFT AND SET UP
185 REM * CONTROL CHARACTERS
190 REM
200 PRINT"⌐"
210 POKE89,CP:POKE90,LP:POKE91.1:POKE92,1
215 POKE786,1:SYS(28960)
220 POKE2016,3:POKE2018.4:POKE2020.9:POKE2022,15
230 GOSUB430
240 REM
250 REM GET CURSOR CONTROL INPUTS
260 REM
270 GETA$:IFA$=""THEN270
280 IFA$="⬛"THENGOSUB560
290 IFA$="▮"THENGOSUB690
300 IFA$="▮"THENGOSUB820
310 IFA$="⌐"THENGOSUB950
320 IFA$="▨"THENGOSUB1080
330 REM
340 REM 'F1' IS CONTROL KEY IF PRESSED
345 REM THEN 'C' ON CONTROL LINE
350 REM IS REVERSED AND CONTROL OPTIONS
355 REM ENABLED. THESE ARE:
360 REM I -INPUT DATA TO SCREEN.
365 REM O -OUTPUT TO PRINTER
370 REM D -DELETE ENTIRE LARGE SCREEN
375 REM CONTENTS
380 REM
390 IFA$="▮"THENPOKE2016.131:GOSUB1170
400 POKE 197,255
```

```
410 GOTO230
420 REM
430 REM DISPLAY CONTROL LINE
440 REM
450 POKE89,0:POKE90,23:POKE91,38:POKE784,67
455 POKE786,7:SYS(28800)
460 POKE87,24:POKE88,0:SYS(29216)
470 PRINT"          COLUMN     LINE          ";
480 POKE87,24:POKE88,14:SYS(29216)
490 PRINTCC;
500 POKE87,24:POKE88,25:SYS(29216)
510 PRINTLC;
515 FORDM=56256TO56295:POKEDM,1:NEXT
520 RETURN
530 REM
540 REM CURSOR CONTROL SUBROUTINES
550 REM
560 REM CURSOR UP
570 REM
580 LC=LC+1:IFLC>23ORLC=23THENL=L+1
590 IFLC>75THENLC=75
600 IFL>50THENL=50
610 POKE89,CP:POKE90,LP:POKE91,1:POKE92,1
615 POKE786,3:SYS(28960)
620 LP=LC
630 IFLC>22THENLP=22
640 POKE825,L
650 SYS(31173)
660 POKE89,CP:POKE90,LP:POKE91,1:POKE92,1
665 POKE786,1:SYS(28960)
670 RETURN
680 REM
690 REM CURSOR LEFT
700 REM
710 CC=CC-1:IFCC>39ORCC=39THENC=C-1
720 IFCC<0THENCC=0
730 IFC<0THENC=0
740 POKE89,CP:POKE90,LP:POKE91,1:POKE92,1
745 POKE786,6:SYS(28960)
750 CP=CC
760 IFCC>39ORCC=39THENCP=39
770 POKE824,C
780 SYS(31173)
790 POKE89,CP:POKE90,LP:POKE91,1:POKE92,1
795 POKE786,6:SYS(28960)
800 RETURN
810 REM
820 REM CURSOR RIGHT
830 REM
840 CC=CC+1:IFCC>40ORCC=40THENC=C+1
850 IFCC>80THENCC=80
```

168

```
860 IFC>40THENC=40
870 POKE89,CP:POKE90,LP:POKE91,1:POKE92,1
875 POKE786,7:SYS(28960)
880 CP=CC
890 IFCC>39ORCC=39THENCP=39
900 POKE824,C
910 SYS(31173)
920 POKE89,CP:POKE90,LP:POKE91,1:POKE92,1
925 POKE786,3:SYS(28960)
930 RETURN
940 REM
950 REM·CURSOR DOWN
960 REM
970 LC=LC-1:IFLC>220RLC=22THENL=L-1
980 IFLC<0THENLC=0
990 IFL<0THENL=0
1000 POKE89,CP:POKE90,LP:POKE91,1:POKE92,1
1005 POKE786,6:SYS(28960)
1010 LP=LC
1020 IFLC>22THENLP=22
1030 POKE825,L
1040 SYS(31173)
1050 POKE89,CP:POKE90,LP:POKE91,1:POKE92,1
1055 POKE786,1:SYS(28960)
1060 RETURN
1070 REM
1080 REM CURSOR HOME
1090 REM
1100 POKE86,CP:POKE87,LP:POKE88,1:POKE89,1
1105 POKE786,1:SYS(28960)
1110 CC=0:C=0:CP=0:POKES24,C
1120 LC=0:L=0:LP=0:POKE825,L
1130 SYS(31173)
1140 POKE89,CP:POKE90,LP:POKE91,1:POKE92,1
1145 POKE786,1:SYS(28960)
1150 RETURN
1160 REM
1170 REM INPUT CONTROL FUNCTION KEY -
1175 REM PRESS 'F1' TO EXIT CONTROL MODE
1180 REM
1190 GOSUB430
1200 GETA$:IFA$=""THEN1200
1210 IFA$="D"THEN1270
1220 IFA$="I"THEN1410
1230 IFA$="O"THEN1750
1240 IFA$<>"▆"THEN1250
1245 POKE2016,3:FORQ=1TO100:NEXT:RETURN
1250 GOTO1190
1260 REM
1270 REM PERFORM LARGE SCREEN ERASE
1275 REM FUNCTION - 'CONTROL' AND 'D'
```

169

```
1280 REM
1290 POKE2018,132
1300 POKE87,24:POKE88,0:SYS(29216)
1310 PRINT" DELETE?";
1320 GETA$:IFA$=""THEN1320
1330 IFA$<>"Y"THEN1370
1340 SYS(31124)
1350 SYS(31173)
1360 GOSUB1080
1370 GOSUB430
1380 POKE2016,3:POKE2018,4
1390 RETURN
1400 REM
1410 REM PERFORM DATA INPUT FUNCTION -
1415 REM 'CONTROL' AND 'I'
1420 REM
1430 POKE2020,137
1440 GOSUB430
1450 POKE197,255:REM REPEAT
1460 GETA$:IFA$=""THEN1460
1470 IFA$="⊠"THENGOSUB560:GOTO1440
1480 IFA$="▌"THENGOSUB690:GOTO1440
1490 IFA$="▐"THENGOSUB820:GOTO1440
1500 IFA$="⊐"THENGOSUB950:GOTO1440
1510 IFA$="▧"THENGOSUB1080:GOTO1440
1520 REM
1530 REM PRESS 'F1' TO EXIT INPUT MODE
1540 REM
1550 IFA$<>"▓"THEN1560
1555 POKE2016,3:POKE2020,9:FORQ=1TO100:NEXT:RETURN
1556 FORQ=1TO100:NEXT:RETURN
1560 POKE87,LP:POKE88,CP:SYS(29216)
1565 POKE646,Z
1570 PRINTA$;
1575 Z=PEEK(646)
1580 SYS(31261)
1590 IFCC=79THEN1620
1600 GOSUB820
1610 GOTO1440
1620 CC=0:CP=0:C=0
1630 POKE824,C
1640 LC=LC+1:IFLC>22THENL=L+1
1650 IFLC>75THENLC=75
1660 IFL>50THENL=50
1670 POKE89,CP:POKE90,LP:POKE91,1:POKE92,1
1675 POKE786,7:SYS(28960)
1680 LP=LC
1690 IFLC>22THENLP=22
1700 POKE825,L
1710 SYS(31173)
1720 POKE99,CP:POKE90,LP:POKE91,1:POKE92,1
```

```
1725 POKE786,7:SYS(28960)
1730 GOTO1440
1740 REM
1750 REM OUTPUT CONTENTS OF LARGE SCREEN
1755 REM TO PRINTER - 'CONTROL' AND 'O'
1760 REM
1770 POKE2022,143
1780 OPEN4,4
1790 FORQ=20480TO26558STEP82
1800 FORX=QTOQ+79
1810 A=PEEK(X)
1820 B=A
1830 IFA<32THENB=A+64:IFA>63THENB=32
1840 A$=CHR$(B)
1850 PRINT#4,A$;
1860 NEXTX
1870 PRINT#4,CHR$(13);
1880 NEXTQ
1890 CLOSE4
1900 POKE2016,3:POKE2022,15
1910 RETURN
READY.
```

SCREEN SAVE AND READ

The second section in the Display Management contains two Basic programs for saving the contents of the screen and reading the Screen Save back onto the screen, using a disk drive (they could be modified for cassette use).

SCREEN SAVE

The Screen Save program first opens a sequential file:
OPEN 2,8,2,"@0:DUMP,S,W"
     The '@' assumes one has already saved the file before. If there is no existing file then remove the '@' symbol. Two loops are set up. The first (1024-1943 in steps of 10) indicates the screen area to be saved to disk. Only the first 23 lines are saved, the last two lines have been reserved for command word entry. In other words on the last two lines one can execute the programs and save the screen contents without saving the commands. The variable A$ is initialised and the second loop (0 to 10) is set up. The next step is to store the contents of the screen into A$ in blocks of 10 and then output it to the disk. Before jumping back to read the next block A$ is again set to a null string. The colour is then output to the disk in exactly the same way using the colour locations for each character on the screen. Lastly the file is closed.

SCREEN READ

The Screen Read routine first sets the cursor on the 23rd line of the screen. The program then opens a sequential file, this is done with OPEN 2,8,2,"DUMP,S,R"; where 'DUMP' is a sample file name, S stands for sequential and R stands for read. Having opened the file the program executes a loop in steps of 10 (1024 to 1943 the first 23 lines), using the array A$ to store the contents of the screen in blocks of 10. The colour is then read from the disk, and the same method used to set the correct colour data onto the screen using the colour memory. Finally, the program closes the file.

```
10000 OPEN2,8,2,"@0:DUMP,S,W"
10010 A$=""
10020 FOR I=1024 TO 1943 STEP 10
10030 FOR J=0 TO 10
10040 A$=A$+STR$(PEEK(I+J))+","
10050 NEXT J
10060 A$=LEFT$(A$,LEN(A$)-1)
10070 PRINT#2,A$
10080 A$="":NEXT I
10090 FOR I=55296 TO 56215 STEP 10
10100 FOR J=0 TO 10
10110 A$=A$+STR$(PEEK(I+J))+","
10120 NEXT J
10130 A$=LEFT$(A$,LEN(A$)-1)
10140 PRINT#2,A$
10150 A$="":NEXT I
10160 CLOSE2
READY.
```


```
10000 PRINT"◤▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨":DIM A(9)
10010 OPEN 2,8,2,"DUMP,S,R"
10020 FOR I=1024 TO 1943 STEP 10
10030 INPUT#2,A(0),A(1),A(2),A(3),A(4),A(5),A(6
),A(7),A(8),A(9)
10040 FOR J=0 TO 9
10050 POKEI+J,A(J)
10060 NEXT J,I
10070 FOR I=55296 TO 56215 STEP 10
10080 INPUT#2,A(0),A(1),A(2),A(3),A(4),A(5),A(6
),A(7),A(8),A(9)
10090 FOR J=0 TO 9
10100 POKEI+J,A(J)
10110 NEXT J,I
10120 CLOSE 2
READY.
```

173

APPENDICES

Screen Display Codes

SET 1	SET 2	POKE	SET 1	SET 2	POKE	SET 1	SET 2	POKE
@		0	[		27	6		54
A	a	1	£		28	7		55
B	b	2	]		29	8		56
C	c	3	↑		30	9		57
D	d	4	←		31	:		58
E	e	5	SPACE		32	;		59
F	f	6	!		33	<		60
G	g	7	"		34	=		61
H	h	8	#		35	>		62
I	i	9	$		36	?		63
J	j	10	%		37	⊟		64
K	k	11	&		38	♠	A	65
L	l	12	'		39	▯	B	66
M	m	13	(		40	⊟	C	67
N	n	14	)		41	⊟	D	68
O	o	15	*		42	⊟	E	69
P	p	16	+		43	▢	F	70
Q	q	17	,		44	▮	G	71
R	r	18	-		45	▯	H	72
S	s	19	.		46	◜	I	73
T	t	20	/		47	◟	J	74
U	u	21	0		48	◞	K	75
V	v	22	1		49	▢	L	76
W	w	23	2		50	◺	M	77
X	x	24	3		51	◿	N	78
Y	y	25	4		52	▢	O	79
Z	z	26	5		53	▢	P	80

SET 1	SET 2	POKE	SET 1	SET 2	POKE	SET 1	SET 2	POKE
●	Q	81	▮		97	⊟		113
▢	R	82	▬		98	⊞		114
♥	S	83	▢		99	⊟		115
▯	T	84	▢		100	▯		116
◿	U	85	▢		101	▮		117
⊠	V	86	▓		102	▯		118
◯	W	87	▢		103	▢		119
♣	X	88	▒		104	▢		120
◫	Y	89	◤	▨	105	▬		121
◆	Z	90	▯		106	▢	☑	122
⊞		91	⊩		107	▛		123
▨		92	▜		108	▙		124
◫		93	⊿		109	▟		125
⊡	▨	94	⌐		110	▙		126
◥	◪	95	▢		111	▚		127
SPACE		96	⌐		112			

Codes from 128-255 are reversed images of codes 0-127.

Text elements in the diagram:

CHARACTER SET
(bit 2 of $0001)
0 = CHAR ROM
1 = I/O ROM/RAM

$FFFF — (65535)

KERNAL ROM

8K HI RAM
(bit 1 of $0001)
0 = RAM
1 = KERNAL ROM

$E000 — (57344)
CIA 1, CIA 2 (S. Bus, P.U.P)
$DC00 — (55320)
Colour Ram Nibbles
$D800 — (55296)
VIC II, SID
$D000 — (53248)

4K RAM
maps to $D000 when
bits 0 & 1 of $0001 = 0

4K RAM

$C000 — (49152)

BASIC
INTERPRETER
ROM

8K LO RAM
(bit 0 of $0001)
0 = RAM
1 = BASIC ROM

$A000 — (40960)

EXROM
8K ROM Cartridge
maps here

$8000
(32768)

BASIC

User RAM

(38912 Bytes)

$4000
(16384)

HI-RES Screen
maps here

$2000
(8192)

VIC II Chip sees
this 16K block
on power-up

$0800 — (2048)
Screen (1K)
$0400 — (1024)
Workspace (1K)
Processor Reg ($0/1)
$0000

HI-RES Colour Table

179

COLOUR	IN LISTINGS	NORMAL	MULTI-COLOUR
BLACK	▓	0	8
WHITE	▟	1	9
RED	▓	2	10
CYAN	◣	3	11
PURPLE	▓	4	12
GREEN	▓	5	13
BLUE	▓	6	14
YELLOW	▓	7	15
ORANGE	▓	8	N/A
BROWN	▓	9	N/A
LT. RED	▓	10	N/A
DK. GREY	▓	11	N/A
MD. GREY	▓	12	N/A
LT. GREEN	▓	13	N/A
LT. BLUE	▓	14	N/A
LT. GREY	▓	15	N/A

APPENDIX C
Colour Chart Codes
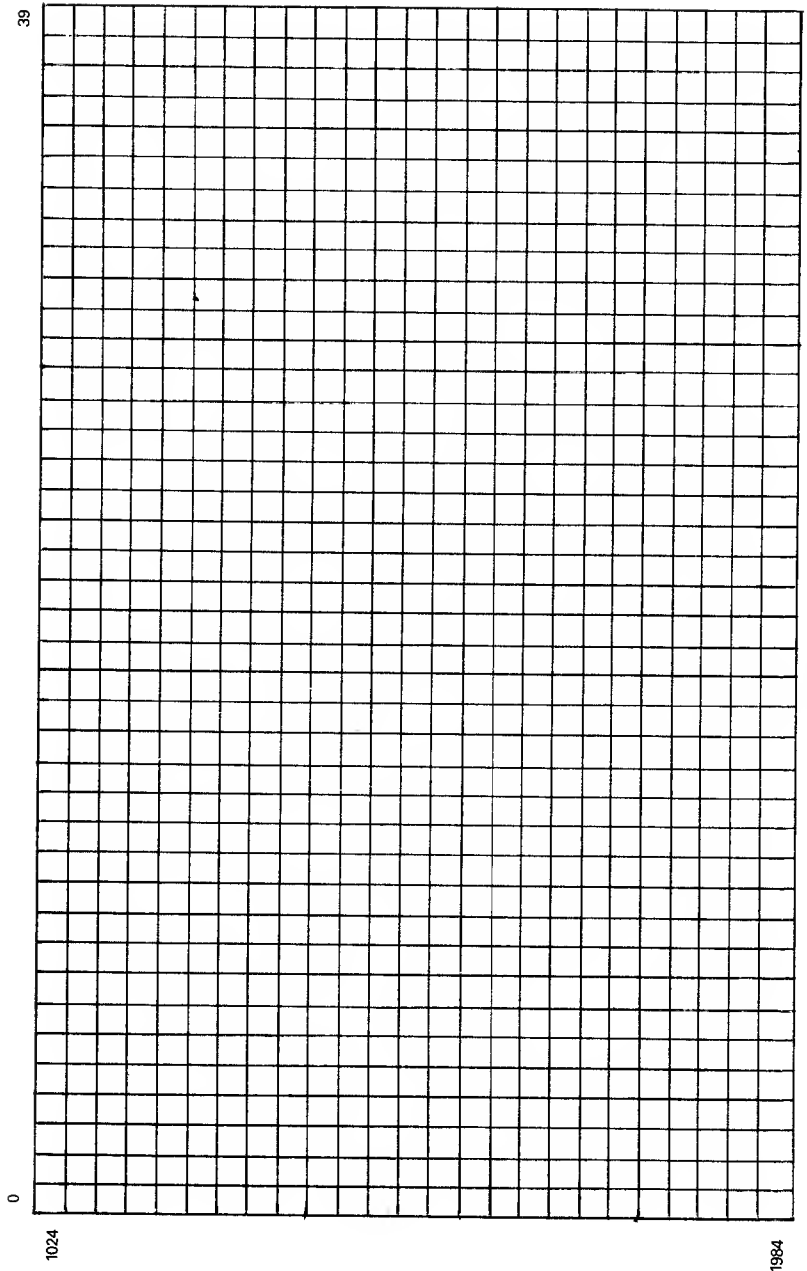
APPENDIX D
Screen memory Grid

181
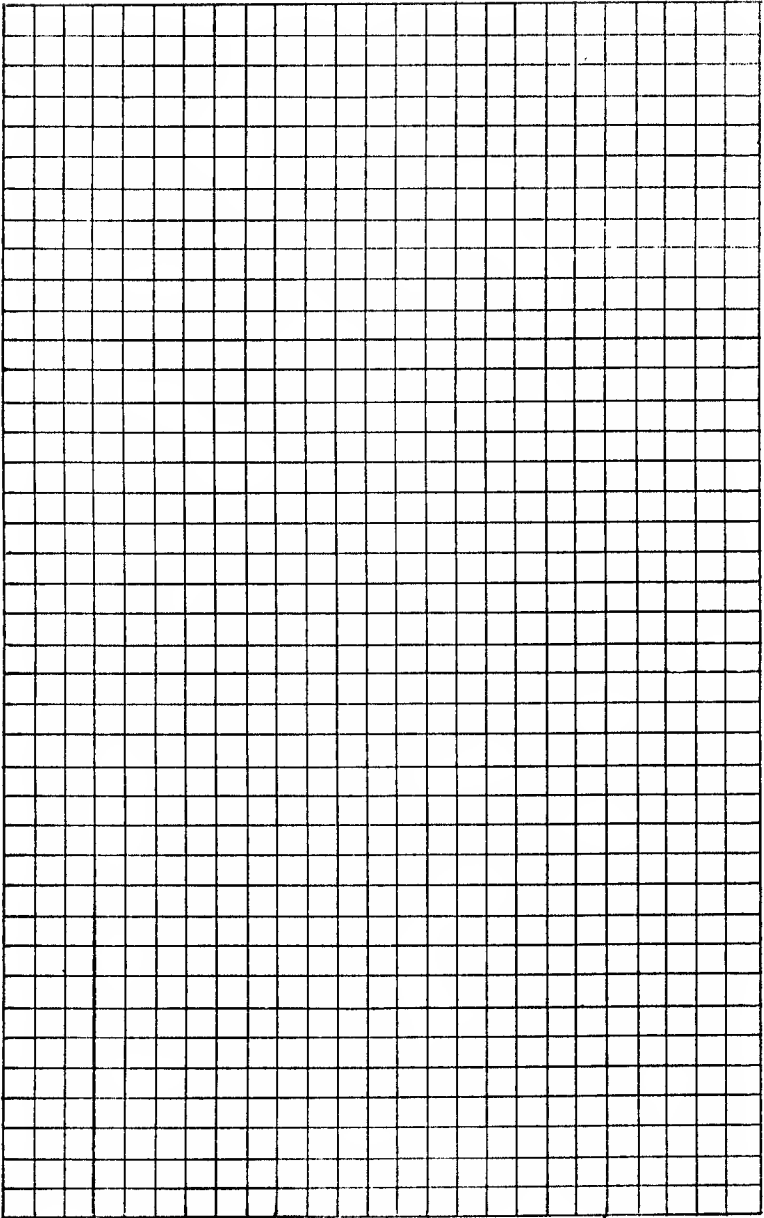
APPENDIX E
Colour memory Grid

39

0

55296

56256

BYTE # 1   BYTE # 2   BYTE # 3

| | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | 12B | 64 | 32 | 16 | 8 | 4 | 2 | 1 | 128 | 64 | 32 | 16 | B | 4 | 2 | 1 |

BYTES 1,2,3
BYTES 4,5,6
BYTES 7,8,9
BYTES 10,11,12
BYTES 13,14,15
BYTES 16,17,1B
BYTES 19,20,21
BYTES 22,23,24
BYTES 25,26,27
BYTES 2B,29,30
BYTES 31,32,33
BYTES 34,35,36
BYTES 37,3B,39
BYTES 40,41,42
BYTES 43,44,45
BYTES 46,47,48
BYTES 49,50,51
BYTES 52,53,54
BYTES 55,56,57
BYTES 58,59,60
BYTES 61,62,63

APPENDIX F
Sprite Grid

```
      128 64 32 16  8  4  2  1
BYTE # 1 ┌──┬──┬──┬──┬──┬──┬──┬──┐
      2 ├──┼──┼──┼──┼──┼──┼──┼──┤
      3 ├──┼──┼──┼──┼──┼──┼──┼──┤
      4 ├──┼──┼──┼──┼──┼──┼──┼──┤
      5 ├──┼──┼──┼──┼──┼──┼──┼──┤
      6 ├──┼──┼──┼──┼──┼──┼──┼──┤
      7 ├──┼──┼──┼──┼──┼──┼──┼──┤
BYTE # 8 └──┴──┴──┴──┴──┴──┴──┴──┘
```

```
      128 64 32 16  8  4  2  1
BYTE # 1 ┌──┬──┬──┬──┬──┬──┬──┬──┐
      2 ├──┼──┼──┼──┼──┼──┼──┼──┤
      3 ├──┼──┼──┼──┼──┼──┼──┼──┤
      4 ├──┼──┼──┼──┼──┼──┼──┼──┤
      5 ├──┼──┼──┼──┼──┼──┼──┼──┤
      6 ├──┼──┼──┼──┼──┼──┼──┼──┤
      7 ├──┼──┼──┼──┼──┼──┼──┼──┤
BYTE # 8 └──┴──┴──┴──┴──┴──┴──┴──┘
```

```
      128 64 32 16  8  4  2  1
BYTE # 1 ┌──┬──┬──┬──┬──┬──┬──┬──┐
      2 ├──┼──┼──┼──┼──┼──┼──┼──┤
      3 ├──┼──┼──┼──┼──┼──┼──┼──┤
      4 ├──┼──┼──┼──┼──┼──┼──┼──┤
      5 ├──┼──┼──┼──┼──┼──┼──┼──┤
      6 ├──┼──┼──┼──┼──┼──┼──┼──┤
      7 ├──┼──┼──┼──┼──┼──┼──┼──┤
BYTE # 8 └──┴──┴──┴──┴──┴──┴──┴──┘
```

APPENDIX G
Character 8x8 Grid

INDEX